Desenvolvimento de componentes para a conceção de células robóticas em ambiente ABB RobotStudio

Carlos Manuel Prada Maia Neto

Dissertação de Mestrado

Orientadores: Professor Paulo Abreu

Professor Manuel Romano Barbosa

Mestrado Integrado em Engenharia Mecânica

Ramo de Automação

Junho de 2016

“Mother, did it need to be so high?”

Pink Floyd

Resumo

Na conceção de células robóticas, em ambiente de programação *off-line*, é muitas vezes necessário dispor de componentes auxiliares tais como garras, mesas posicionadoras, tapetes transportadores, cortinas de luz, e outros componentes para que possa ser possível efetuar simulações realistas da programação do robô e da sua interação com esses equipamentos.

No *software* de programação RobotStudio® da ABB® é possível a utilização de *Smart Components*. Esta funcionalidade permite criar objetos, com determinadas características geométricas e funcionais, que podem ser programados para interagir com a programação do robô. É assim possível criar componentes, com características paramétricas, que podem ser facilmente configuráveis de modo a adaptá-los às necessidades da simulação.

Nesta dissertação foram modeladas duas séries de garras da Schunk® (PGN e PZN) de modo a que possam ser escolhidos e configurados alguns parâmetros de uma forma expedita. Além do componente geométrico criado, estas garras estão providas de um comportamento lógico e possuem funcionalidades que permitem a integração fácil numa célula robótica para que posteriormente possam ser programadas em linguagem RAPID, no RobotStudio®.

Além da criação das duas séries de garras, foi também elaborada uma estação robótica para a verificação e validação das capacidades dos *Smart Components*. Nesta estação foi desenvolvido um programa em linguagem RAPID, tirando partido da funcionalidade do RobotStudio de permitir a comunicação com um controlador virtual, simulando assim o comportamento pretendido num ambiente virtual, o mais próximo possível da célula real.

Development of smart components for the conception of robotic cells in ABB RobotStudio

Abstract

In the design of robotic cells for off-line programming, it is necessary to have auxiliary components such as grippers, positioning tables, conveyors, light curtains, and others in order to simulate the robot programming and its interaction with these equipments realistically.

In the RobotStudio® programming software it is possible to use Smart Components. This functionality allows the user to create objects with defined properties, both geometrical and functional, that can be used in the robotic cell. In this way it is possible to create components, with parametric properties, that can easily be configured to adapt to the simulation requirements.

In this project, two series of Schunk® grippers (PGN and PZN) were created as Smart Components with certain parameters that can be chosen and configured. These grippers are provided with a logic behaviour and with functionalities that allow them to be easily integrated in a robotic cell and programmed afterwards in RAPID programming language.

Moreover, a concept cell was created to illustrate the use of the grippers and others Smart Components. In this station, a routine was written in the virtual controller, simulating the intended behaviour in this virtual environment.

Agradecimentos

Aos meus orientadores, Professor Paulo Abreu e Professor Manuel Romano Barbosa, pela disponibilidade, paciência e interesse mostrados ao longo do projeto desenvolvido.

Ao professor Fernando Gomes de Almeida, pela preparação dada aos alunos de dissertação.

Ao Sr. Joaquim e ao Sr. Ramalho, pela companhia e boa disposição.

À minha família, pelo apoio que me presenteou durante todos estes anos, e por serem a principal razão de ter chegado a onde estou.

A todos os meus amigos, por serem sempre o que sempre precisei que fossem.

Obrigado.

Índice de Conteúdos

Índice de figuras

Índice de tabelas

1 Introdução

Nicola Tesla tentou definir um dos maiores sonhos do Homem da seguinte forma: "*I conceived the idea of constructing an automaton which would mechanically represent me, and which would respond, as I do myself, but, of course, in a more primitive manner, to external influences...*" [1]. O sonho era, à altura, construir um máquina obediente e incansável, capaz de realizar as tarefas mais repetitivas e enfadonhas. Desde então, a robótica tem vindo a tentar cumprir este sonho, contribuindo para a crescente complexidade de funções que sabemos hoje ser possível automatizar. Este progresso tem sido feito nas últimas décadas a um ritmo estonteante, chegando hoje a uma situação em que soluções robóticas dão resposta a solicitações previamente de impensável execução pelo ser humano. Poderia até dizer-se que fomos, enquanto espécie, ultrapassados pela nossa própria criação. Aliás, vivemos num mundo em que é já raro encontrar processos fabris sem elementos automáticos, substituindo o trabalho humano sempre que possível. E para substituir tão complexa máquina como o ser humano, a robótica teve de ter um crescimento tão grande quanto a sua procura.

Ocorre que este argumento está no cerne de uma problemática ética relativa à previsão que a crescente importância da Máquina tenha por consequência a decrescente importância do Homem. E a esta questão é necessário dar o progresso como resposta. É importante recordar que a Revolução Industrial foi um dos mais importantes acontecimentos na história da Humanidade, o que é facilmente sustentado pelo impacto que teve na evolução da população mundial:

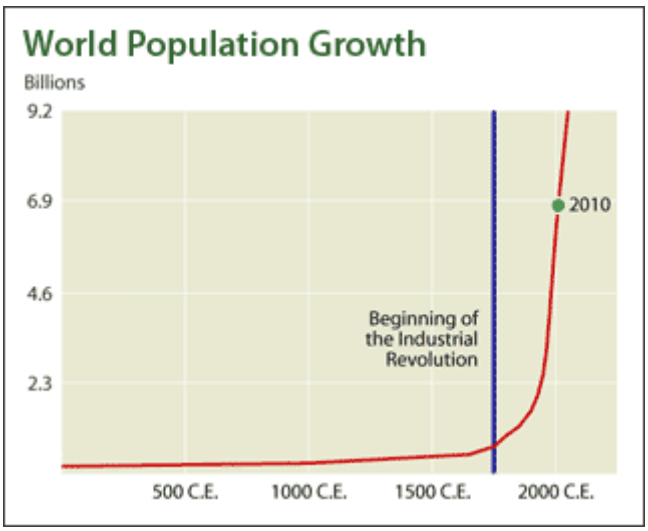


Figura 1.1 – Crescimento da população mundial [2]

É essencial manter a consciência do contributo positivo que este desenvolvimento tecnológico tem tido para o Homem, e a robótica industrial vem no seu seguimento,

libertando os operadores fabris de tarefas puramente físicas para funções criadoras de maior valor acrescentado; aumentando a produtividade dos processos e reduzindo por isso o custo dos produtos fabricados; e em última instância elevando a fasquia do possível ao apresentar novos processos e possibilidades a quem se atreve a empurrar os limites do conhecimento. Não somos, assim, ultrapassados pela nossa criação, mas fazemos dela uma ferramenta integrante de nós próprios no caminho do progresso e evolução.

Por tudo isto é de tão grande importância o trabalho desenvolvido neste âmbito, particularmente pela relevância que tem hoje em dia o dimensionamento, conceção e simulação de sistemas robotizados.

1.1 Enquadramento e motivação

A indústria atual procura, cada vez mais, soluções para a automatização de processos com vista à melhoria da produtividade. Numa área onde o crescimento tem sido grande e se projeta que continue a ser, como é possível observar pelos dados presentes na figura 1.2 [3], onde o número de robôs vendidos tem sido sempre crescente, com a notória exceção do ano de 2009 devido à conhecida crise económica, são sempre relevantes os estudos que incidem sobre esta temática.

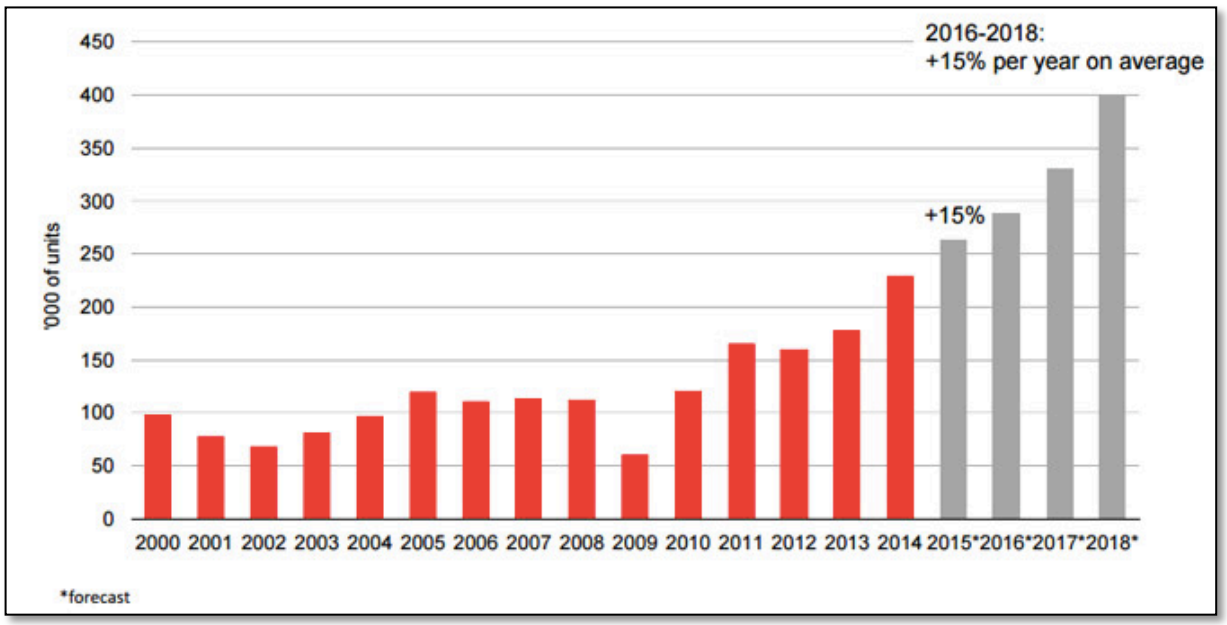


Figura 1.2 - Procura anual de robôs industriais 2000-2018 [3]

Para responder a esta necessidade, as empresas produtoras destas células robóticas, oferecem também software de programação para que possa ser feita a configuração dos robôs. Esta programação é bastante importante para testar qualquer sistema que possa ser implementado nas linhas de produção, com vista a optimizar o mesmo antes de ele ser fisicamente montado.

Uma dessas empresas é a ABB (Asea Brown Boveri), que é lider de mercado na produção de robôs a nível mundial [4], e que disponibiliza o seu software ABB RobotStudio para que os seus robôs sejam programados quer *online*, quer em *off-line*.

A programação *online* era a tradicionalmente usada e consistia em configurar os movimentos/trajetórias do robô através da consola ligada ao controlador. Ainda que este tipo tenha algumas vantagens face à programação *off-line*, como não necessitar de um operador especialmente qualificado para a realizar, hoje em dia é adoptada em grande parte a programação e simulação das células robóticas com recurso a *software*. A programação *off-line* permite assim um aumento de produtividade, pois não implica qualquer paragem nas linhas de montagem e aumenta também a segurança para os operadores.

Sendo que a programação *online* pode ser desenvolvida por operários menos qualificados, procura-se facilitar a tarefa a quem, de forma a conseguir aumentar a produtividade da sua linha de produção, opta por construir um sistema virtual, sem que para isso sejam necessários conhecimentos avançados de programação e um exagerado dispêndio de tempo, que é radicalmente reduzido com a criação dos *Smart Components*.

Esta dissertação tem por base este crescimento de importância dos softwares de programação *off-line* de robôs, sendo que o desafio foi o de tornar o processo de criação de uma célula robótica mais expedito.

1.2 Objetivos

O principal objetivo desta dissertação é o de explorar a utilização das funcionalidades do RobotStudio na criação de *Smart Components*. A finalidade dos *Smart Components* é tornar mais expedita a conceção e programação de células robóticas. Os *Smart Components* a desenvolver deverão permitir a escolha e configuração de garras pneumáticas, que são um exemplo típico de componentes a integrar em células robóticas de operações de manipulação.

Os *Smart Components* devem também ser dotados de funcionalidades lógicas que permitam a sua interligação com o controlador de uma célula robótica de forma expedita, sendo fácil a sua integração e configuração no programa da célula e do robô.

1.3 Estrutura

Além da introdução, em que é feita uma primeira apresentação do projeto, bem como um resumo dos objetivos e um enquadramento, este relatório conta ainda com mais 4 capítulos.

O segundo capítulo, **RobotStudio® e *Smart Components***, visa explicar a definição de SC, tanto no software em que o trabalho foi desenvolvido, como noutros em que também é utilizado. Também são dados alguns exemplos de SC já existentes na biblioteca *online* da ABB e é justificado o seu interesse.

No terceiro capítulo, **Desenvolvimento e conceção dos *Smart Components*,** é descrito o processo realizado para a criação dos Smart Components relativos a garras e apresentadas as suas funcionalidades.

O quarto capítulo, **Integração de *Smart Components* em célula robótica**, aborda uma possível utilização dos Smart Components criados no capítulo anterior, de forma a justificar a sua utilidade e a capacidade de tornar o processo de programação/configuração da célula robótica mais expedito.

Por último, o quinto capítulo, **Conclusões e perspectivas de trabalho futuro**, em que se faz uma análise global ao trabalho desenvolvido, bem como uma reflexão sobre as limitações encontradas, dificuldades sentidas no desenvolvimento do projeto, e ainda sobre possíveis trabalhos futuros que possam acrescentar algo a este projeto.

O documento faz-se acompanhar por dois anexos.

2 RobotStudio® e *Smart Components*

Neste capítulo é feita uma abordagem inicial ao RobotStudio e aos *Smart Components (SC)*, com o objetivo de apresentar o *software*, indicar as suas principais capacidades e definir o conceito de *Smart Component*.

2.1 RobotStudio®

O RobotStudio é um *software* que permite a modelação, a programação *off-line* e a simulação de células robóticas. É distribuído por uma das maiores empresas dedicadas à indústria robótica, a ABB, e é por isso usado exclusivamente para fazer a programação e simulação dos robôs da empresa.

Uma das grandes vantagens da programação *off-line* é a de poder simular processos reais de uma estação robótica sem ter de os implementar fisicamente e essa é uma das principais capacidades do RobotStudio, já que este software dá a oportunidade ao utilizador de trabalhar com um controlador, o IRC5, que corre virtualmente no *software*. Este controlador virtual realiza as operações de cinemática inversa que permitem a movimentação (*Jog*) do robô, garantindo ou não, a aplicabilidade do programa simulado no sistema real.

O *software* permite selecionar, de entre os vários robôs disponibilizados pela ABB, o modelo a utilizar na simulação de uma célula robótica. O RobotStudio também disponibiliza uma biblioteca de equipamentos normalmente usados na construção de células robóticas, como é o caso dos tapetes transportadores, paletes, garras, barreiras, algumas ferramentas para acoplar aos robôs que permitem simular operações de pintura (pistolas) ou soldadura (tochas), etc. Alguns destes equipamentos são simplesmente componentes geométricos (paletes, tapetes), enquanto que outros têm já incorporado uma lógica comportamental associada (garras).

Outra das vantagens do RobotStudio é a de estar em constante desenvolvimento, quer seja pela própria ABB, empresa criadora e distribuidora do software que disponibiliza frequentemente atualizações, quer pela comunidade de utilizadores que partilham informação e aplicações desenvolvidas através da página *web* da empresa.

Na figura 2.1 é visível o ambiente de trabalho do software, aquando da abertura de uma estação previamente definida.

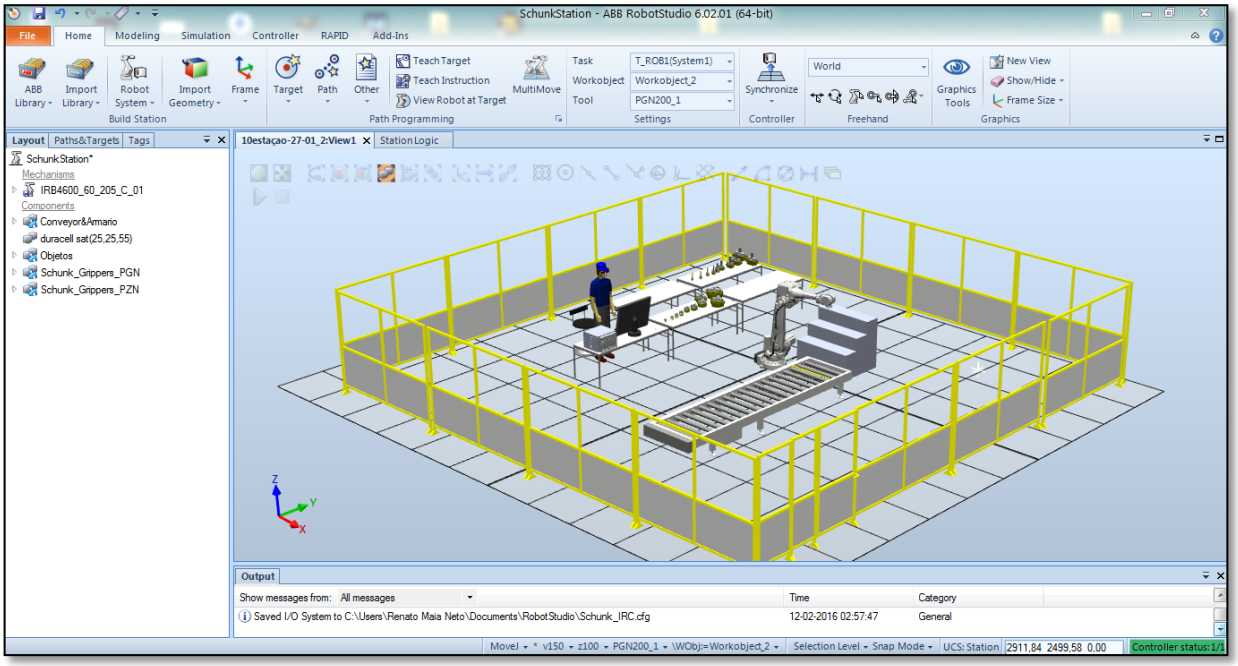


Figura 2.1 - Ambiente de trabalho do RobotStudio

O RobotStudio possui um manual de utilização bastante completo. Aliado aos menus de ajuda para realizar operações específicas no próprio software e aos exemplos disponibilizados pela ABB, constituem de facto um conjunto de ferramentas bastante úteis no desenvolvimento de aplicações, tornando assim também o *software* alvo de projetos como este.

Uma listagem não exaustiva das funcionalidades do software RobotStudio inclui:

- Importação de ficheiros CAD
- Criação objetos CAD (ACIS)
- Geração automática de trajetórias
- Deteção de colisões
- Verificação de alcances
- Aplicabilidade na célula real
- Integração com um controlador virtual
- Criação de *Smart Components* (SC)
- Criação de *Add-ins*

2.2 Smart Components

Não é recente a tentativa de criar soluções capazes de facilitar o trabalho com softwares específicos de engenharia. À medida que vão aumentando as suas capacidades, aumenta também a complexidade da sua utilização, e é apenas natural e expectável a promoção de instrumentos que possam, de uma maneira adequada, evidenciar estas capacidades.

A definição de *Smart Components* no âmbito do RobotStudio é a de um objeto, com ou sem representação gráfica, que tem um comportamento estabelecido que pode ser implementado através de programação em C# ou por agregação de um SC de base no próprio RobotStudio.

Os SC podem ser criados de duas maneiras distintas: ou por programação direta recorrendo ao *software* "Microsoft Visual Studio", em linguagem de programação C#, ou por integração dos SC de base usando a interface disponibilizada no ambiente RobotStudio.

Noutras aplicações são também usados os conceitos de SC, ainda que possam em alguns casos não ser referidas como tal. Nos *browsers* como o "Mozilla Firefox" ou o "Google Chrome*"*, em *software* de *design* como o "Adobe Photoshop*"* ou o "Adobe Publisher" ou em software CAD como o "AutoDesk 3d-Studio Max*"* são usadas extensões que permitem adicionar funcionalidades com base em comportamentos que podem ser parametrizados, sejam este criados em formato Active-X, Java ou .NET. Apesar de poderem não ser referidos como SC, estas extensões possuem um conceito similar.

A tabela 2.1, presente no manual RobotStudio [5], descreve as diferentes terminologias associadas a um SC.

Tabela 2.1 – Terminologias de um *Smart Component* [5]

Termo	Definição
Code Behind	Uma classe .NET associada a um *SC* que pode implementar um determinado comportamento reagindo a certos eventos.
[Dynamic] *Property*	Um objeto associado a um SC que tem valor, tipo e outras características. O valor da propriedade é usado pelo código para controlar o comportamento do SC.
[Property] *Binding*	Liga o valor de uma propriedade ao valor de outra propriedade.
[Property] *Attributes*	Valores-chave que contêm informação adicional acerca de cada propriedade dinâmica.
[I/O] *Signal*	Um objeto associado a um SC que tem valor e direção (*input/output*) análogo a sinais de I/O num controlador.
[I/O] *Connection*	Liga o valor de um sinal ao valor de outro sinal
Aggregation	O processo de ligar vários SC usando *bindings* e/ou ligações de modo a implementar um comportamento mais complexo
Assets	Objeto de informação contido num SC. É usado para localizar recursos contidos no código.

A classe *Code Behind* permite associar ao SC um comportamento lógico criado noutro software, em linguagem de programação C#. Alternativamente a este, existe o editor de SC no próprio software, onde se pode criar, editar e aglomerar SC usando uma interface gráfica. É possível, através da criação destas propriedades e das conexões entre SC que o RobotStudio disponibiliza, criar as lógicas de comportamento pretendidas para as aplicações desenvolvidas e assim fazer com que, de uma forma mais expedita, sejam criados componentes para integração na simulação de células robóticas.

Na criação de SC através do RobotStudio, as classes mais usadas são as Propriedades e os Sinais. Através da construção destas duas classes e das interligações realizadas entre os SC primitivos, é possível criar lógicas complexas, sem que seja necessária a escrita em código e compilação noutro *software*.

A última classe, *Assets,* faz com que seja possível adicionar recursos a um SC. Um exemplo da implementação de um na criação de um SC é a garra desenvolvida por Richard Ramos [6] disponibilizada na galeria ABB *online*: sempre que é dada a ordem de fecho, faz com que seja emitido um sinal sonoro. O ficheiro que contém o sinal sonoro está presente nesta classe. Pode também ser usado para acrescentar uma série de ficheiros que sirvam de tutorial ao uso dos SC, do tipo PDF, ou ainda alguns dados numéricos sobre a estação, em ficheiro Excel.

2.2.1 Criação de *Smart Components* por programação em C#

A ABB disponibiliza três SDK (*Software Development Kit*) [7] com os correspondentes "API's" (*Application Programming Interface*) que permitem o desenvolvimento de aplicações através de programação em C#. São eles o "PC-SDK", "RobotStudio-SDK" e o "FlexPendant-SDK".

O RobotStudio-SDK é o necessário para criar as aplicações para o *software*. Nele estão contidas bibliotecas de instruções que permitem ao utilizador alargar as funcionalidades já existentes e personalizá-las conforme as especificações pretendidas. É com este SDK que são desenvolvidos os SC por programação em C#. O "PC-SDK" serve para fazer as ligações ao controlador, tanto ao real como ao virtual. O "FlexPendant-SDK" é o que permite criar interfaces e aplicações para a consola de controlo dos robôs.

Na definição de um SC por este método são utilizados dois conjuntos de códigos distintos. Em XML (eXtensible Markup Language), são declaradas as variáveis de propriedades e sinais necessárias ao programa. Na figura 2.2, é possível observar um exemplo de um código em XML de um SC *ObjectCompare* [8] em que são declaradas duas variáveis dinâmicas e um sinal de *output*.

```
XML

<lc:Library fileName="ObjectCompare.rslib">

<SmartComponent icon="ObjectCompare.png"
    codeBehind="ObjectCompare.ObjectCompare">
        <Properties>
          <DynamicProperty name="ObjectA"
                           valueType="ABB.Robotics.RobotStudio.ProjectObject"/>
          <DynamicProperty name="ObjectB"
                           valueType="ABB.Robotics.RobotStudio.ProjectObject"/>
        </Properties>
        <Signals>
          <IOSignal name="Output" signalType="DigitalOutput" readOnly="true"/>
        </Signals>
   </SmartComponent>
  </lc:Library>
```

Figura 2.2 – Código XML *ObjectCompare*

O outro código criado presente no *template* fornecido, é escrito em C# e tem como objetivo implementar a resposta aos eventos criados pelas variáveis declaradas em XML, como é possível observar na figura 2.3.

```
public override void OnPropertyValueChanged(SmartComponent component,
            DynamicProperty changedProperty, object oldValue)
{
    #region ObjectComparer1
    ProjectObject a = (ProjectObject)component.Properties["ObjectA"].Value;
    #endregion

    #region ObjectComparer2
    ProjectObject b = (ProjectObject)component.Properties["ObjectB"].Value;
    #endregion

    #region ObjectComparer3
    component.IOSignals["Output"].Value = a == b ? 1 : 0;
    #endregion
}
```

Figura 2.3 – Código C# *ObjectCompare*

A ABB disponibiliza bibliotecas de funções para que possam ser desenvolvidas as mais diversas aplicações através de C#, no entanto existe pouca informação e exemplos para o estudo mais aprofundado deste tipo de SC, o que torna menos acessível a sua criação através de código. Este processo oferece uma flexibilidade elevada na criação de SC, no entanto requer conhecimentos elevados, muito específicos da área da programação de aplicações.

2.2.2 Criação de *Smart Components* por integração de elementos SC base

O RobotStudio apresenta um conjunto de SC, que são definidos como os de base, já que têm como objetivo executar funções lógicas simples e, por isso, são normalmente usados como parte de SC mais complexos.

O conjunto de SC de base encontram-se estruturados nas seguintes grupos: “Sinais e Propriedades”, “Primitivas Paramétricas”, “Sensores”, “Ações”, “Manipuladores” e “Outros”.

Em “Sinais e Propriedades”, está disponível uma série de SC que possibilitam construir operações e expressões lógicas através da ligação destas com os sinais e as propriedades implementadas no SC. É possível também converter variáveis analógicas em variáveis digitais, e vice-versa. São também disponibilizados temporizadores, sinais de relógio que podem ser usados para definir intervalos na simulação e fazer a sua paragem. É possível ver na figura 2.4, um SC *Latch* que serve para implementar o comportamento de *Set/Reset.*

Figura 2.4 – *Smart Component LogicSRLatch*

Nas "Primitivas paramétricas", existe uma série de componentes que são criados através das propriedades que lhes são fornecidas, daí serem definidos como parametrizáveis. É possível criar modelos geométricos de um paralelepípedo, como é possível ver na figura 2.5, ou de um cilindro. Além destes, existe a possibilidade de fazer uma extrusão a uma face de um componente gráfico ou multiplicar um componente geométrico tridimensionalmente.

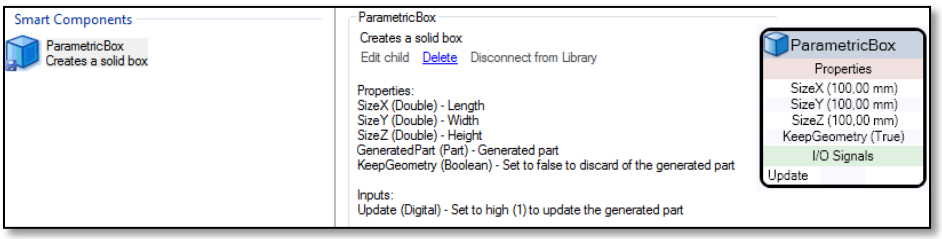


Figura 2.5 - *Smart Component ParametricBox*

Nos "Sensores" estão presentes os SC que fornecem ao utilizador a informação das interseções entre objetos presentes na estação. Sensores de colisão e sensores de linha, como se observa na figura 2.6, são alguns dos componentes deste grupo. Estes sensores são geralmente definidos por vectores, através de dois pontos, e dispõem de um sinal que comuta quando algum objeto é interceptado no volume definido.

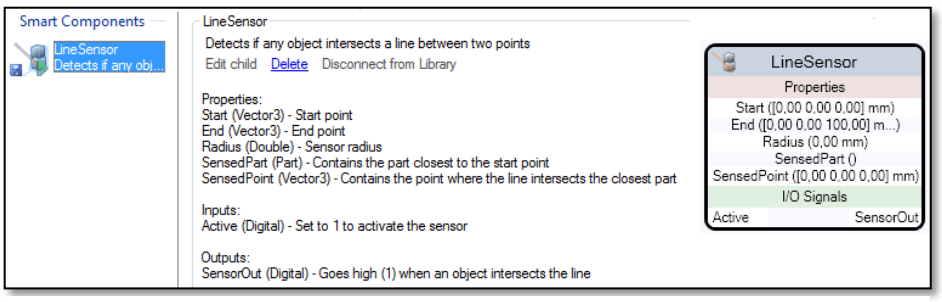


Figura 2.6 – *Smart Component LineSensor*

No grupo "Ações" existe a possibilidade de fazer ligações físicas entre objetos gráficos, como é o caso do SC *Attacher* (figura 2.7), e do SC *Dettacher*, que se comporta de forma inversa. Os restantes SC presentes neste grupo servem para mostrar, remover e multiplicar objetos gráficos.

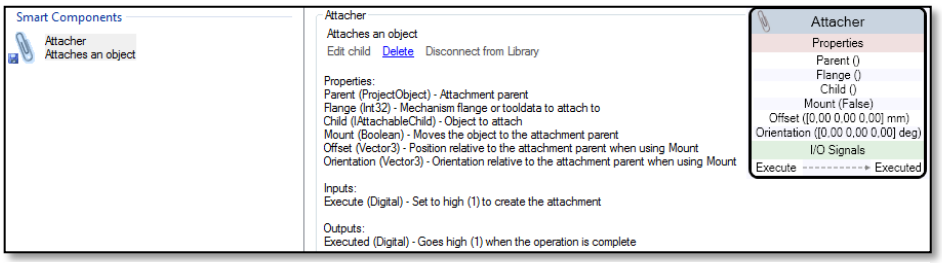


Figura 2.7 – *Smart Component Attacher*

Nos "Manipuladores", estão representados os SC que pretendem recriar um movimento. Este movimento pode ser descrito por um objeto e uma dada trajetória, ou pela mudança de configuração das juntas de um mecanismo previamente definido, como é o caso representado na figura 2.8.

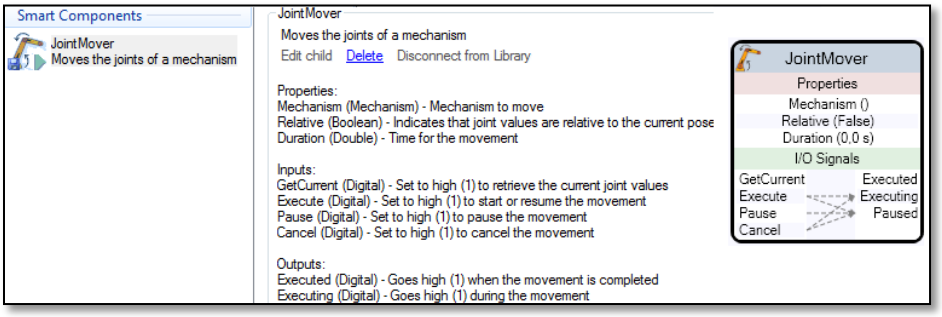


Figura 2.8 – *Smart Component JointMover*

Em "Outros", existe uma série de SC que não se encaixa nas restantes categorias e que permite ao utilizador emitir um sinal, gerar um número aleatório, gerar um sinal pulsado durante a simulação, enviar uma mensagem de erro/aviso ou mudar a cor de um objeto. Na figura 2.9 está representado o *Logger* que envia, quando executado, uma mensagem de informação/erro/aviso pré-definida para a caixa de mensagens do RobotStudio.

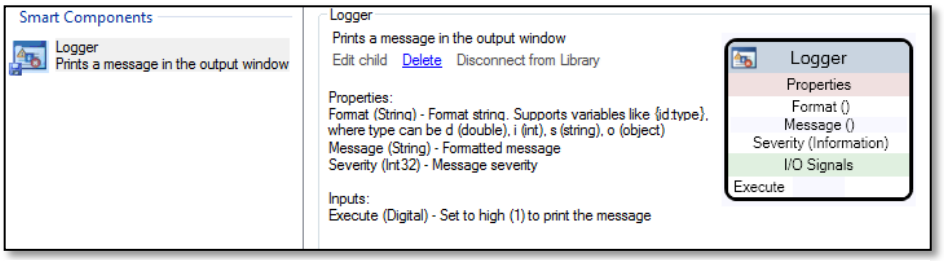


Figure 2.9 – *Smart Component Logger*

Para criar um SC por integração dos elementos de base, o utilizador tem disponíveis certas funcionalidades no software RobotStudio, devendo proceder do modo ilustrado no seguinte exemplo:

1- iniciar o processo de criação de um SC;

2- adicionar um componente, quer seja um componente de base ou um modelo geométrico de um objeto;

3- definir as propriedades e as ligações (*bindings*) entre estes;

4- definir os sinais e as ligações (*connections*) a atribuir.

A aba de design do RobotStudio permite uma visualização gráfica da estrutura do SC. Nesta estrutura estão incluídos os mecanismos presentes, as ligações internas, as propriedades e as suas ligações.

Como pode ser observado na figura 2.10, a aba de design permite ao utilizador fazer as ligações (4) entre as propriedades (1) e os sinais de I/O (2 e 3) criados no SC e os componentes de base que são importados. Inúmeras lógicas de comportamento podem ser criadas com a implementação dos sinais/propriedades e das suas conexões aos SC de base, sendo por isso esta visualização gráfica uma opção viável para o desenvolvimento de um SC. No entanto, a necessidade de fazer a importação de demasiados componentes e mecanismos pode tornar o processo pouco adequado pela reduzida área de visualização e, nesse caso, é preferível realizar as ligações através dos menus "*Properties and Bindings*" e "*Signals and Connections*".

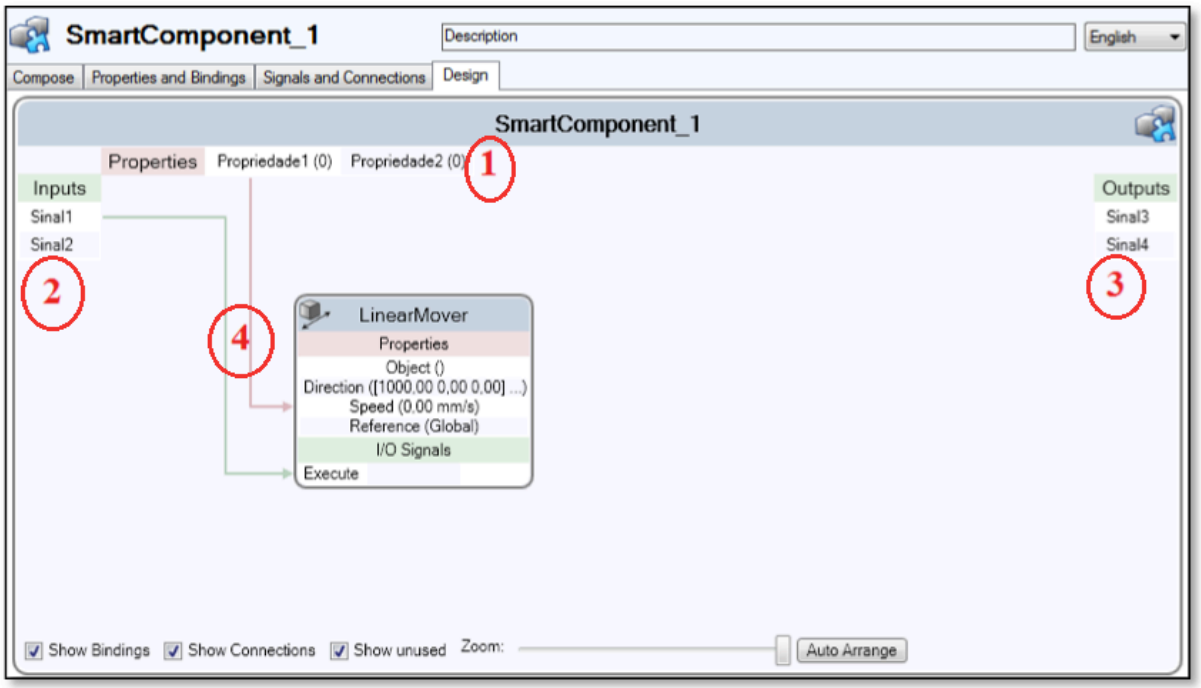


Figure 2.10 - Aba de *design* de um SC

2.3 Exemplos de *Smart Components*

Através dá página *online* da ABB [9], numa galeria própria de partilha de criações desenvolvidas por utilizadores é possível ter acesso a cerca de três dezenas de SC. Uma análise a esses SC disponibilizados permitiu classificar e agrupá-los nos três grupos seguintes:

- **Componentes para células**

  Estes componentes servem para ajudar a tornar as células robóticas mais realistas, portanto apresentam uma série de objetos que são normalmente usados aquando da construção das mesmas.

  Exemplos: Pedestais para os robôs, barreiras de segurança, paletes

- **Animação e simulação de processos**

  Na simulação criada no RobotStudio, podem ser usadas algumas ferramentas que proporcionam ao utilizador um ambiente mais imersivo.

  Exemplos: Câmara em movimento, partículas de soldadura

- **Mecanismos**

  Por último, os mecanismos pretendem facilitar ao utilizador uma série de componentes, parametrizáveis e possivelmente com lógica comportamental através de sinais, que podem ser usados para recriar o ambiente pretendido.

  Exemplos: Mesas posicionadoras, garras, tapetes transportadores, máquina CNC, semáforos, cilindros pneumáticos

Além de serem divididos nestas categorias, os SC presentes nesta biblioteca são discriminados entre componentes e mecanismos, aquando da importação para o software. Esta diferenciação é feita pela presença ou ausência de uma lógica associada ao seu comportamento. No caso dos que são considerados componentes, observa-se esse comportamento, contrariamente ao mecanismo que apenas tem uma cinemática definida.

Como exemplo de SC da biblioteca *online* da ABB, existe uma barreira paramétrica, criada por Hope Kong [10], que permite, ao introduzir o comprimento dos lados da barreira, criar um objeto geométrico com as medidas pretendidas. Como pode ser observado na figura 2.11, existe um pequeno menu onde são introduzidas as propriedades e um botão "Update" que cria a barreira. A barreira criada é somente um objeto gráfico, não tendo associado a ele qualquer comportamento lógico.

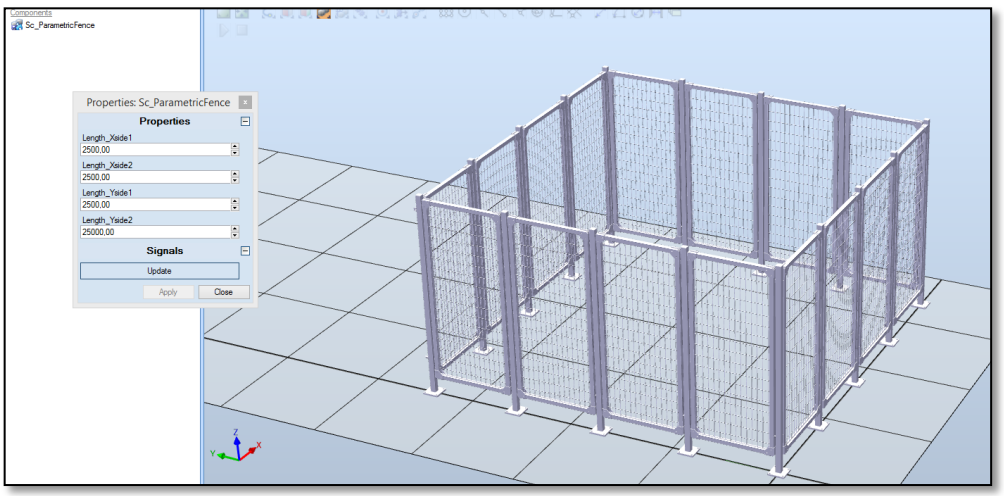


Figura 2.11 – Barreira Paramétrica

Outro exemplo de SC, representado na figura 2.12, é a garra de dois dedos da Schunk, de Richard Ramos [6]. Este SC permite a abertura e fecho dos dedos da garra através de um sinal de *input* "diGripper". Contém também um SC *LineSensor,* representado a amarelo, que comuta um sinal quando alguma peça se encontra no espaço de trabalho da garra. Este SC já tem uma lógica comportamental associada porque, além do componente geométrico, tem sinais de I/O que podem ser configurados.

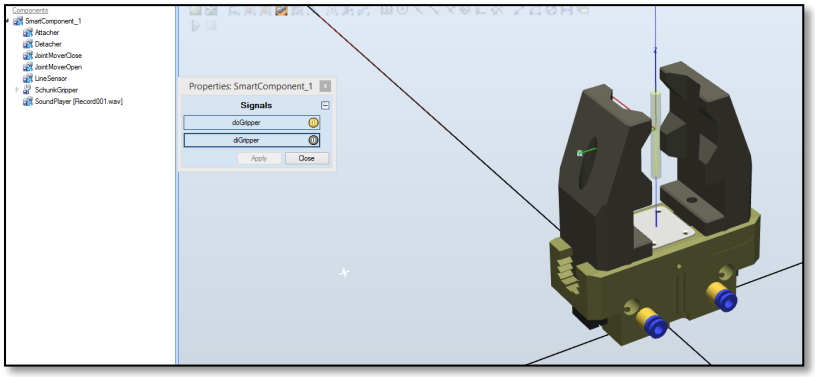


Figura 2.12 – Garra Schunk

Um dos poucos trabalhos apresentados à comunidade devidamente fundamentado sobre o desenvolvimento de um SC e que serviu como mote à criação deste projeto é o de Joseph Kopacz [11]. Neste projeto é desenvolvido um conjunto de componentes que permitem tornar expedita a criação de uma estação de soldadura. O projeto foi desenvolvido para a empresa Wolf Robotics e foram disponibilizados no site da ABB os SC individuais criados.

Cada SC desenvolvido é constituído por três componentes. O primeiro é uma barreira paramétrica que é construída na estação, sendo que o comprimento e a largura são definidos pelo utilizador. O segundo é o construtor da trajetória, que cria a mesa com os dados que são inseridos para o comprimento e largura. Por último é criada uma cortina de luzes virtual entre dois postes que delimitam o espaço da estação. É possível observar na figura 2.13, a célula criada, através da introdução de alguns parâmetros, como o comprimento e largura pretendidos da mesa. Toda esta lógica de comportamento é programada com "Microsoft Visual Studio".

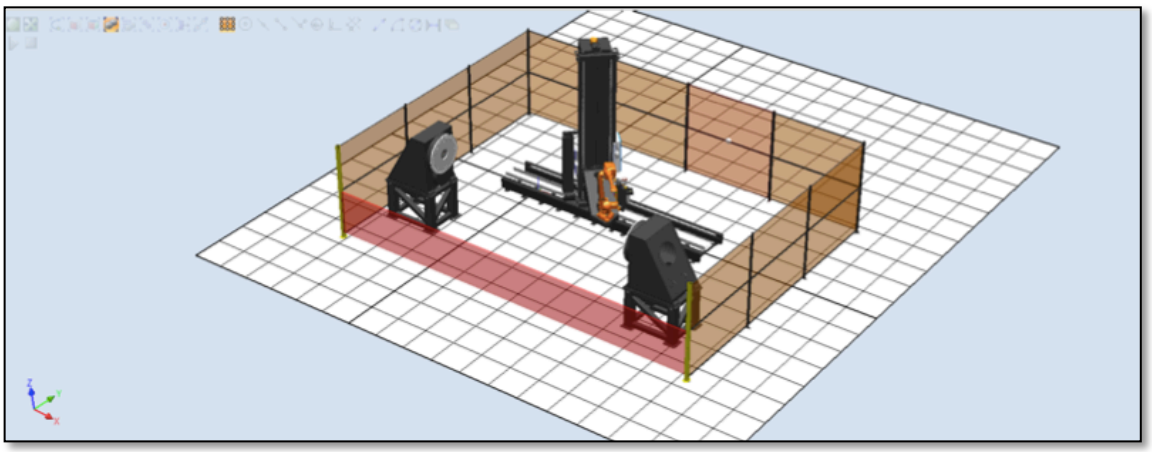


Figura 2.13 – Célula criada pela Wolf Robotics através de SC

3 Desenvolvimento e conceção dos *Smart Components*

Neste capítulo será descrita a escolha dos SC a serem modelados e será apresentado o procedimento de desenvolvimento e criação dos mesmos.

3.1 Seleção dos componentes a modelar

Estudada a oferta de soluções existente na página *online* da ABB e no *software* RobotStudio, optou-se pela criação de um SC que pudesse funcionar como uma biblioteca de uma série de garras. Desta forma, seria dada ao utilizador a possibilidade de escolher um único elemento e configurar um conjunto de parâmetros que têm de ser definidos aquando da escolha de uma garra, proporcionando assim uma criação e aplicação expedita de uma garra na célula robótica. Associado a cada um dos modelos geométricos destas garras é implementada uma lógica comportamental, através de propriedades e sinais de I/O. Isto permite ao utilizador aceder a uma série de variáveis auxiliares que facilitam a programação posterior da célula, sendo assim somente necessário ligar ao controlador estas variáveis implementadas na garra.

As garras escolhidas para modelação foram de duas séries, ambas da Schunk. A série PGN, uma garra de acionamento pneumático de dois dedos paralelos e a série PZN, uma garra também de acionamento pneumático, mas de três dedos concêntricos. Foram escolhidas estas duas séries de garras, tendo como principais motivos a variedade de aplicações em que estas soluções podem ser aplicadas e o facto de serem as garras comercialmente mais procuradas [12]. Tendo em conta que, na indústria, em particular na de produção em massa, se utilizam garras para aplicações bastante específicas, normalmente até com uma aplicação singular, torna-se importante dar a possibilidade de poder criar uma garra que corresponda aos parâmetros pré-definidos pelo utilizador.

3.1.1 Série PGN

A série de garras PGN é uma série de dois dedos paralelos, destinada ao manuseamento de objetos de pequenas dimensões com elevada precisão. Dentro de várias aplicações, é possível observar um exemplo na figura 3.1, em que uma garra PGN se encontra a retirar um objeto após este ser maquinado.

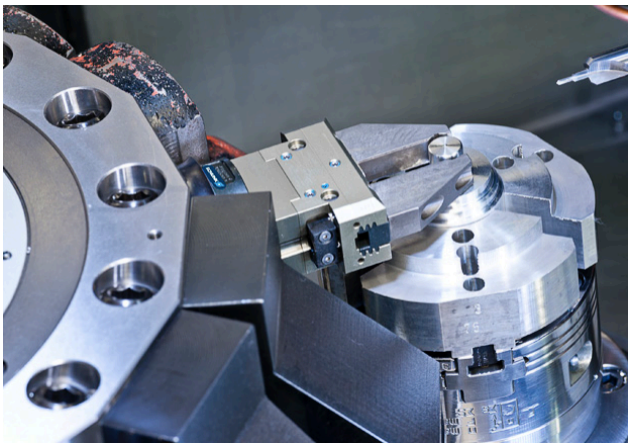


Figura 3.1 Garra PGN em funcionamento

Há também a possibilidade deste tipo de garras em vez de serem acopladas a um robô , serem usadas numa solução integrada. É o caso ilustrado na figura 3.2, com uma unidade de *Pick and Place* constituída por um módulo linear em cada eixo, que permite fazer a movimentação da garra.

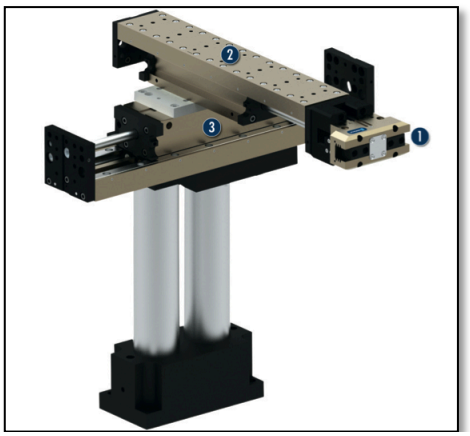


Figura 3.2 – Unidade *Pick And Place*

O facto de esta garra ser a mais procurada de todas as disponibilizadas pela Schunk pode ser explicado pela vasta gama de aplicações em que é aplicável, devido à amplitude de dimensões das peças e dos pesos suportados. Assim, a possibilidade de configurar o comprimento de abertura da garra, aliada à possibilidade de conceber uma solução em que o manuseamento é feito por fora (aplicação tradicional) ou por dentro (no caso da garra operar no interior de um orifício, por exemplo), aumenta a flexibilidade das garras e o seu espectro de aplicações.

Além da capacidade óbvia de abertura e fecho da garra para agarrar o objeto, a Schunk providencia para cada tipo de garra uma série de acessórios do mais variado tipo, como sensores de proximidade indutivos, medidores de força, sensores de posição analógica, sistemas de mudança automática de ferramenta, unidades de compensação de tolerâncias ou pratos de adaptação para a ligação das garras ao robô. Nenhum destes acessórios foi

fisicamente implementado na garra modelada. No entanto, em termos de simulação e para facilitar a programação foram criados alguns sinais que seriam a representação virtual destes mesmos sensores.

O funcionamento destas garras é facilmente explicado através das figuras 3.3 e 3.4, presentes no manual de utilização da série PGN [13]. A garra funciona como um cilindro pneumático de duplo efeito. Quando a câmara principal não está pressurizada, como se observa na figura 3.3, o êmbolo está recuado o que faz com que os dedos das garras estejam fechados.

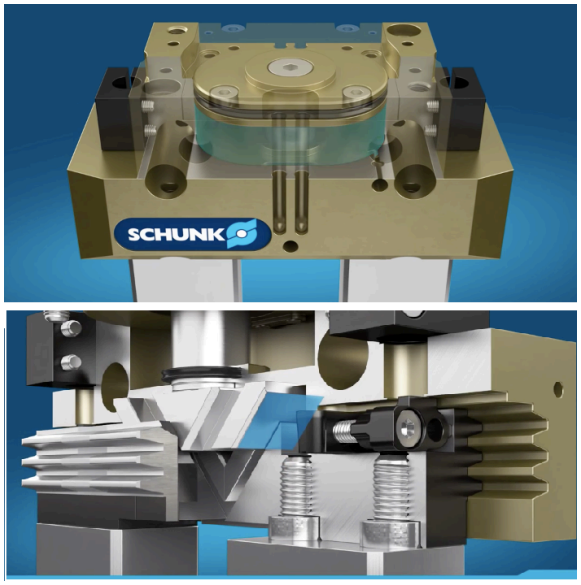


Figura 3.3 – Garra fechada

Quando a eletroválvula direcional que comanda o cilindro é comutada de forma a pressurizar a câmara principal (figura 3.4), o êmbolo movimenta-se de forma a empurrar uma cunha triangular que garante que a abertura dos dedos da garra seja realizada, sendo que cada dedo se movimenta de igual forma, mas com sentido inverso.

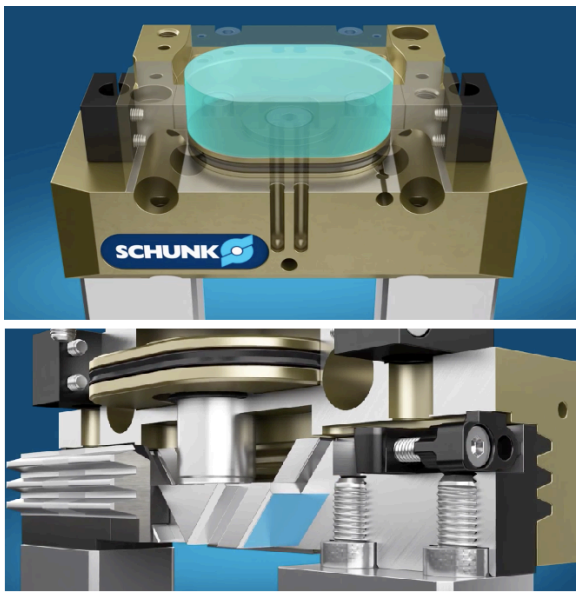


Figura 3.4 – Garra aberta

Para poder transportar este comportamento para o *software*, foi definido que as garras seriam comandadas através de uma electroválvula direcional pneumática 4/2, como se pode ver no circuito pneumático presente na figura 3.5. São necessários dois sinais para comandar a garra, um para a abertura e um para o fecho. A escolha de uma válvula biestável deve-se ao facto de ser uma opção mais segura, em caso de falha de energia elétrica.

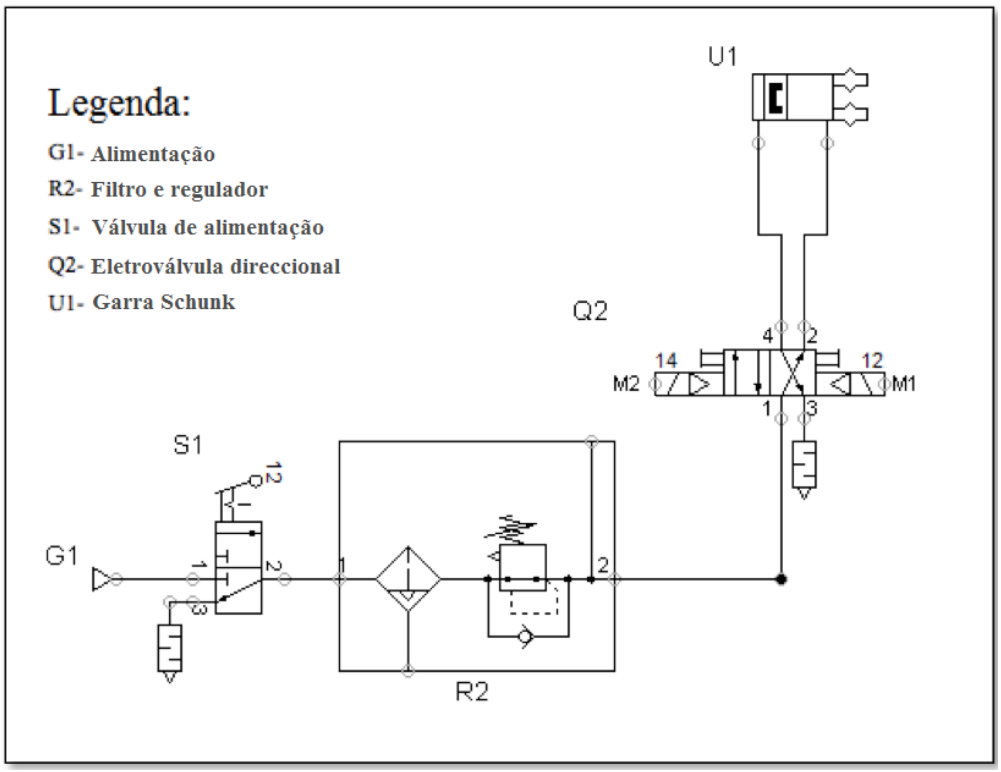


Figura 3.5 – Circuito pneumático

Os sinais a serem replicados no RobotStudio para controlo da garra são os solenoides que executam a ordem de abertura (M1) e de fecho (M2). No RobotStudio será criada também a lógica de *Set/Reset* através de uma *Latch,* evitando desta forma que os dois solenoides possam estar ativos em simultâneo. A válvula S1 é uma válvula de alimentação que não terá representação no *software*.

Para ser possível a parametrização de um conjunto de garras é preciso primeiro definir quais as características que podem ser alvo de debate aquando da escolha de uma garra. Não podendo fazer variar no RobotStudio todas as propriedades de cada garra, foram escolhidas aquelas que era possível recriar no *software*.

Na tabela 3.1 podem ser vistas as três características de cada garra da série PGN que foram usadas para parametrizar a garra: o curso linear por dedo, o peso recomendado máximo para cada peça e o comprimento máximo dos dedos. Isto permite-nos balizar o SC das garras criado para peças relativamente pequenas, já que, com este conjunto de garras, é possível pegar em objetos até 70 mm de largura. O peso máximo possível é dado pela maior garra da série (PGN300), 30 kg. O comprimento máximo do conjunto, 350 mm, também corresponde ao da maior garra. A título de exemplo é apresentado no Anexo B o *datasheet* da garra PGN-plus 40.

Tabela 3.1 – Características das garras PGN-plus

Garras	Curso linear por dedo (mm)	Massa máxima da peça (kg)	Comprimento máximo dos dedos (mm)
PGN-plus 40	2.5	0.62	58
PGN-plus 50	4	0.7	72
PGN-plus 64	6	1.25	90
PGN-plus 80	8	2.1	110
PGN-plus 100	10	3.3	145
PGN-plus 125	13	5.4	180
PGN-plus 160	16	8.2	220
PGN-plus 200	25	13.5	280
PGN-plus 240	30	21.5	320
PGN-plus 300	35	30	350

3.1.2 Série PZN

A série de garras PZN da Schunk é de três dedos concêntricos, como é visível na figura 3.6. Foi escolhida esta série já que o RobotStudio permite a criação de dois tipos de geometrias parametrizáveis (paralelepípedos e cilindros). Desta forma, com a série PZN que é normalmente usada para manusear objetos cilíndricos, é possível alargar a possibilidade de utilização do SC.

A configuração dos dedos destas garras teve como princípio o mesmo que foi usado nas garras PGN: com a garra fechada, os dedos permitiriam o mínimo de espaço possível entre eles, estando muito próximos do contacto. Assim, a abertura da garra permite agarrar uma peça cilíndrica, cujo raio seja inferior ao curso linear dos dedos. O valor máximo do raio permitido para as peças cilíndricas para que possam ser agarradas por uma garra PZN é 35mm (Tabela 3.2).

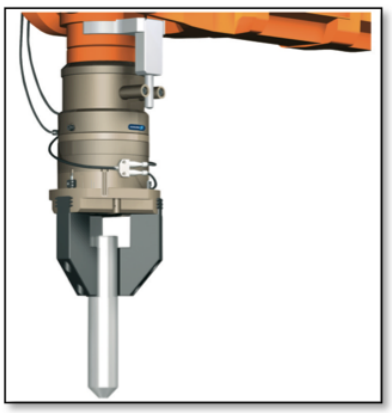


Figura 3.6 – Garra PZN

Na tabela 3.2, é apresentado o resumo das características usadas para a construção de cada garra [14]. A titulo de exemplo, no Anexo B, é apresentado a *datasheet* de uma garra PZN-plus 40.

Tabela 3.2 –Características das garras PZN

Garras	Curso linear por dedo (mm)	Massa máxima da peça (kg)	Comprimento máximo dos dedos (mm)
PZN-plus 40	2.5	1.30	58
PZN-plus 50	4	1.65	72
PZN-plus 64	6	2.9	90
PZN-plus 80	8	3	110
PZN-plus 100	10	9	145
PZN-plus 125	13	15.5	180
PZN-plus 160	16	30	220
PZN-plus 200	25	35	280
PZN-plus 240	30	50	300
PZN-plus 300	35	72.5	250

3.2 Procedimento

O procedimento seguido (figura 3.7) para a criação dos SC das garras é, em primeiro lugar, descrito resumidamente. De seguida, com o exemplo da criação de uma garra, é descrito aprofundadamente, passo a passo. A garra apresentada é uma da série PGN e sendo que a construção das garras PZN é análoga e bastante semelhante, não se justifica repetir a apresentação.

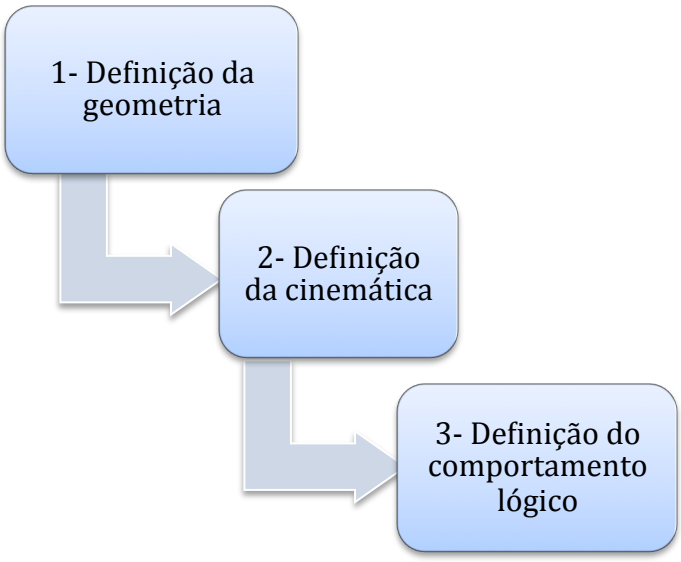


Figura 3.7 – Estágios de criação do *Smart Component* criado

1) Definição da geometria

Um pré-requisito fundamental para a construção de um SC que contenha um modelo gráfico é a importação e integração da geometria no ambiente de programação.

Existem variadíssimos programas de desenvolvimento de modelos 3D e cada um tem, normalmente, o seu formato particular. O RobotStudio permite a importação de vários tipos de ficheiros CAD, sendo que o formato preferencial é o "SAT", porque não obriga a nenhuma conversão. Ainda assim, existe um conversor de formatos embebido no programa em que é possível converter formatos como o "IGES", "STEP", "VDA-FS", "CATIA V4 e V5", "Creo" e "Inventor".

2) Definição do mecanismo

Após a importação das partes criadas, procedeu-se à transformação dessas geometrias em objetos do RobotStudio, com um conjunto de cinemáticas associadas.

3) Definição da lógica comportamental

Por fim, a lógica de comportamento é criada na aba de criação de SC do RobotStudio, tendo os mecanismos criados anteriormente servido como base.

Foram criados dois SC, um para cada série de garras. Esses serão os SC parentais e foram apelidados de "Schunk_Gripper_PGN" e "Schunk_Gripper_PZN". Dentro de cada um desses SC, como se observa na figura 3.8, estão os SC das garras, SC_PGNXX e SC_PZNXX, que estão ligados às propriedades e sinais dos SC parentais para que estes possam ter correspondência e atualizar os valores inseridos pelo utilizador.

Os SC "pais" têm como finalidade servir de interface para a escolha e configuração do SC de cada garra que se pretende utilizar na estação, sendo que os SC "pais" são constituídos por todos os SC de cada garra, conforme pretende evidenciar a figura 3.8.

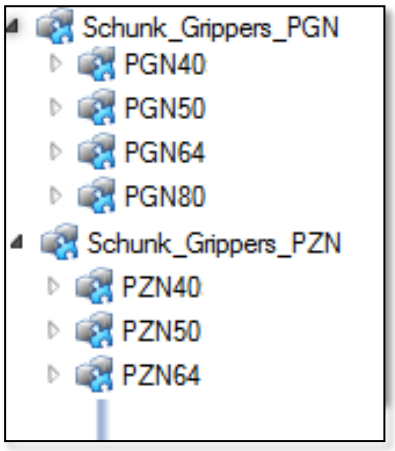


Figura 3.8 – Smart Components constituintes

3.2.1 Definição da geometria

Antes de as garras serem importadas para o RobotStudio, foi necessária uma etapa prévia usando o software de CAD "SolidWorks". Primeiramente, as garras e os dedos foram importados para este programa e foram efetuados os *assemblys* necessários. Na figura 3.9 é possível visualizar a garra PGN40, aquando da importação para o software "SolidWorks".

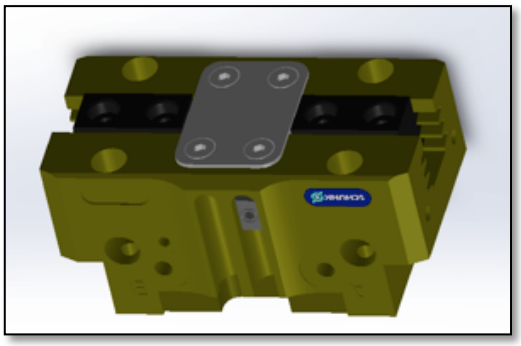


Figura 3.9 – Base da garra PGN

No "SolidWorks", para fazer o *assembly* dos dedos à garra, é preciso definir a posição e a geometria dos dedos. Os dedos disponibilizados pela Schunk são *standard*, com a finalidade de serem posteriormente maquinados para terem a forma pretendida. O objetivo deste SC seria o de possibilitar a parametrização da largura dos objetos de trabalho com que cada garra pudesse trabalhar. Por isso foi definida nesta etapa a configuração dos dedos que seriam montados nas garras PGN.

Foi definido que a configuração dos dedos seria tal que, na situação de ter a garra fechada, a distância entre os dedos fosse praticamente zero. Esta configuração pode ser observada na figura 3.10.

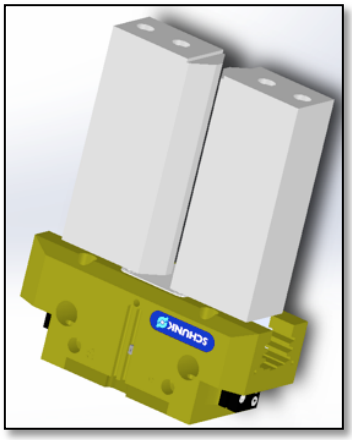


Figura 3.10 – *Assembly* da PGN no SolidWorks

Com esta configuração dos dedos, a largura máxima das peças que podem ser abrangidas para cada garra será de duas vezes o curso dos dedos da mesma. Será assim possível, com a série de garras PGN, pegar em peças com uma largura máxima de 70mm.

Para que a próxima etapa de criar o mecanismo seja possível, cada garra é gravada em 3 ficheiros individuais (“SAT”), um com a base da garra e os outros dois com os respetivos dedos.

3.2.2 Definição da cinemática

Nesta etapa é definida a cinemática da garra. Até então, para o RobotStudio, os ficheiros importados não são mais que modelos geométricos. Como foi referido anteriormente, são importados para cada garra três modelos individuais SAT, para que seja possível criar a cinemática do mecanismo.

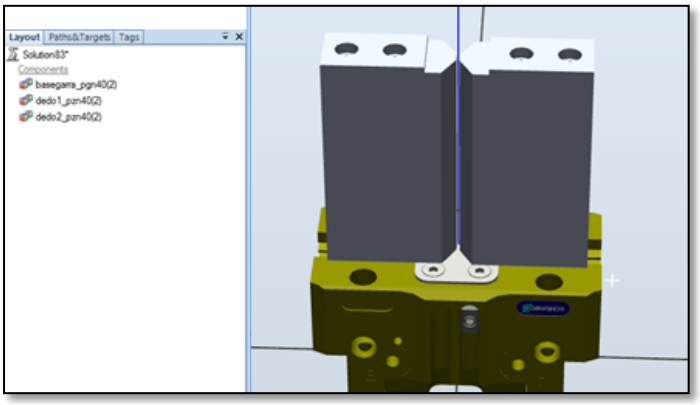


Figura 3.11 – Importação dos ficheiros CAD para o RobotStudio

Para criar o mecanismo, é necessário percorrer uma série de etapas, de forma a definir uma série de propriedades inerentes à ferramenta. Na tabela 3.3 estão representadas as propriedades que têm de ser definidas para a criação de um mecanismo do tipo ferramenta no RobotStudio.

Tabela 3.3 – Submenus da criação de um mecanismo

Submenu	
Links	Os *Links* são o conjunto de modelos geométricos usados para definição da ferramenta. Neste caso, para cada garra existem três Links, sendo que a base da garra é definido como *BaseLink,* já que o movimento dos dedos é realizado em relação a ele.
Joints	A cinemática do mecanismo é criada neste submenu. São definidos os eixos ao longo dos quais os dedos se movem e a sua deslocação máxima.
Tool Data	Sendo este mecanismo uma ferramenta, terá de ter associado a ele uma informação do TCP (*Tool Center Point*), para mais tarde poder ser usada na programação.
Calibration	Este submenu não é usado na criação da ferramenta.
Dependencies	Sendo que o movimento dos dedos não é independente, é neste submenu que se define a dependência. A Joint2 mover-se-á sempre solidária com a Joint1, mas com sentido oposto.

Os parâmetros que constituem o mecanismo desenvolvido são descritos na figura 3.12.

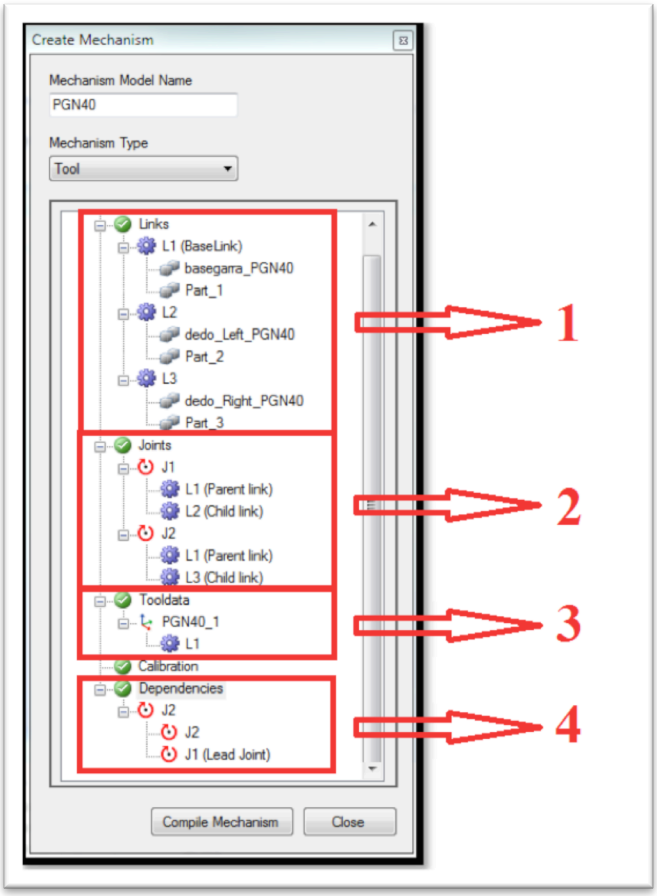


Figura 3.12 – Definição dos parâmetros para a criação do mecanismo

1 - Menu *Links*: A importação individual dos três modelos geométricos que formam a garra deve-se ao facto de, no menu *Links,* estes possuírem movimentações próprias, ainda que obviamente inter-relacionadas.

O corpo da garra é definido como *BaseLink*, já que funciona como charneira ao deslocamento dos dois dedos. O movimento dos dedos será sempre executado em relação à base, permitindo assim à garra ser posicionada de várias formas.

Dentro de cada *Link*, é possível observar que estão alocadas duas geometrias. Uma é naturalmente a geometria correspondente de cada peça, e a outra é um ficheiro "Part_X". Só desta forma o software permite que seja criada uma lógica no SC e que lhe seja adicionada a cinemática desse mesmo *Link*, o que é necessário para a configuração dos dedos das garras, como veremos posteriormente.

2 - Menu *Joints*: É neste menu que são definidos os movimentos dos dedos. São para isso criadas duas *Joints* e para cada uma delas é definido o valor máximo do curso de cada dedo assim como o eixo sobre o qual é feito. Estes valores estão previamente definidos na tabela 3.1 (página 19) para as garras PGN e tabela 3.2 (página 20) para as garras PZN.

3 - Menu *Tooldata*: Neste menu é definido o TCP do mecanismo (figura 3.13). Este ponto ficará guardado como o ponto de serviço na programação aquando da movimentação da garra. Neste menu também é definido o peso da ferramenta, os momentos de inércia e o centro de gravidade, apesar de estes dados serem apenas informativos para esta aplicação, não tendo nenhuma interferência no funcionamento.

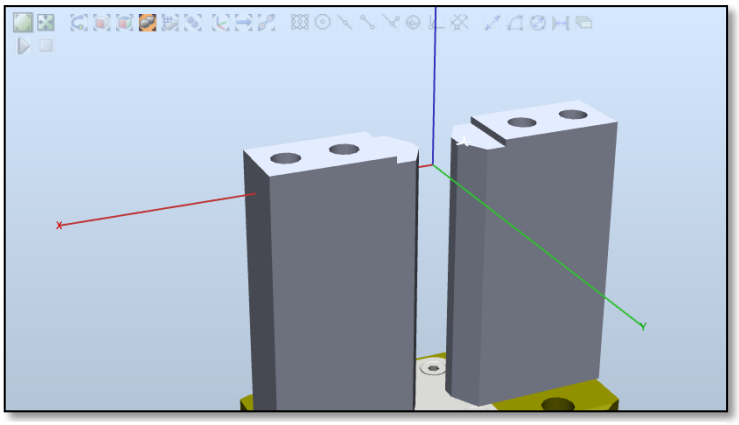


Figura 3.13 – TCP do mecanismo

4 - Menu *Dependencies*: O funcionamento dos dedos, como anteriormente foi descrito, é feito de forma solidária, isto é, cada dedo movimenta-se exatamente de igual forma, só que com sentido inverso. É neste menu que esta lógica é implementada, sendo escolhida uma *Joint* como referência, que pode ser qualquer uma, e a restante como seguidora. No software, a instrução de movimento no sentido contrário é definido através do factor "-1".

Após o mecanismo ser compilado, deixamos de ter um conjunto de modelos geométricos e passamos a ter uma ferramenta com a cinemática descrita criada. Neste momento não é possível fazer mais do que movimentar os dedos da garra entre as posições previamente estabelecidas. Não está instalado ainda qualquer comportamento "inteligente" associado a esta ferramenta.

A figura 3.14 ilustra a manipulação manual dos dedos da garra, fazendo mover o cursor entre as posições mínima e máxima. Apesar de nesta fase ser possível aos dedos permanecerem numa posição intermédia, isso é algo que não acontece assim que é construído o SC, pois só se pretende dotar a garra de duas posições: aberta e fechada.

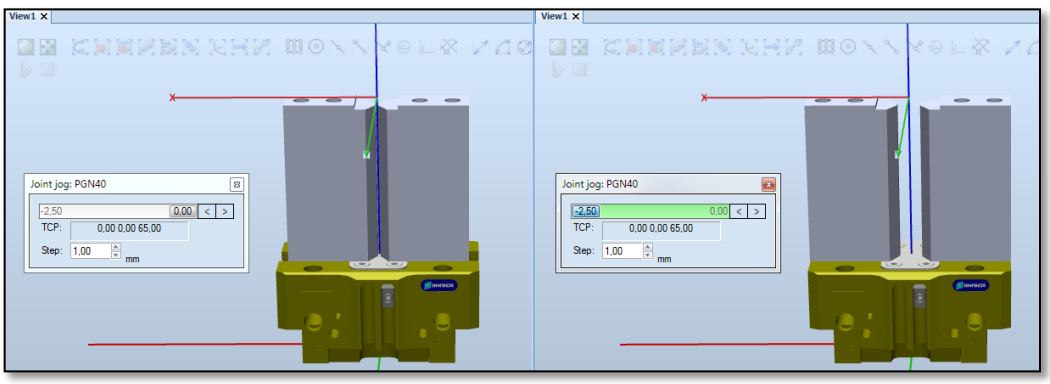


Figura 3.14 – Movimentação (*Jog*) das juntas do mecanismo

3.2.3 Definição da lógica comportamental

A implementação da lógica comportamental nestes SC é o que permite tornar expedito o processo de escolha e configuração das garras criadas, assim como criar funcionalidades que tornem possível a interligação destes SC com o controlador de uma célula robótica.

O SC desenvolvido tanto para a família de garras PGN e PZN da Schunk deve permitir dispor das funcionalidades presentes no esquema da figura 4.15.

Funcionalidades dos SC	Seleção do modelo específico do SC da garra
	Configuração e validação do comprimento dos dedos da garra
	Seleção do robô no qual a garra vai ser acoplada
	Validação da adequação da garra para a manipulação da peça de determinada massa
	Disponibilização de sensores de presença
	Disponibilização de sensores de colisão (garra/peça)
	Disponibilização de sinais de comando de abertura/fecho da garra

Figura 4.15 – Funcionalidades dos SC

Cada uma destas capacidades tem uma lógica associada que teve de ser implementada, tal como descrito nas próximas páginas. Todos estes comportamentos foram criados tanto para o SC da série PGN como para o da série PZN.

Grande parte destas funcionalidades foram criadas usando as funcionalidades disponibilizadas na aba de design dos SC no RobotStudio, onde podem ser estabelecidas as ligações entre as propriedades e sinais dos SC primitivos e destes ao SC criado. Para a explicação das várias lógicas implementadas, são apresentadas imagens simplificadas desta aba para mostrar apenas a lógica da capacidade do SC em análise. A parte dos SC e respetivas ligações que estariam presentes nesse espaço mas não envolvidos nesta análise foram retirados para facilitar a compreensão.

Seleção do modelo específico do SC da garra

O objetivo da criação deste SC é o de proporcionar ao utilizador uma alternativa mais expedita de utilização da família de garras PGN. Para isso, as dez garras da série PGN criadas foram aglomeradas num só SC que serve de interface para a escolha da garra e das características. Neste SC pode ser escolhida a garra a utilizar, dentro da família de garras PGN, como se pode observar na figura 3.16.

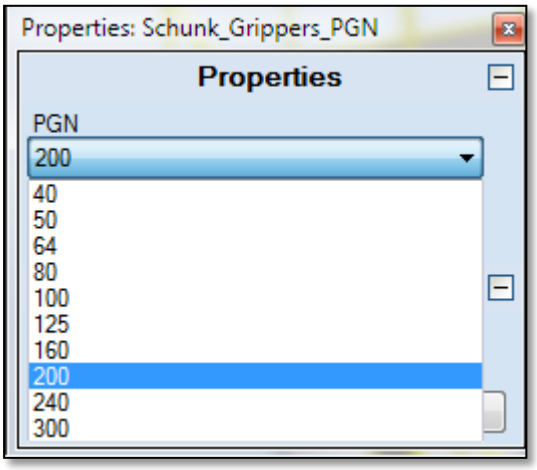


Figura 3.16- Número de série das garras PGN

Esta lógica é implementada no software através da criação de uma propriedade dinâmica, do tipo numérica, cujos valores apresentados correspondem aos números de série das garras, facilitando a identificação.

Como é observável na figura 3.17, é usado um SC *Comparer* (1) com um valor previamente estabelecido igual ao número de série da garra, neste caso 40, juntamente com uma operação lógica SC *AND* (2). Isto permite que quando a garra escolhida tiver o número igual ao do comparador e o sinal "Colocar" seja ativado, seja executado um SC *Attacher* (3) que faz com que haja um acoplamento entre a garra e o robô.

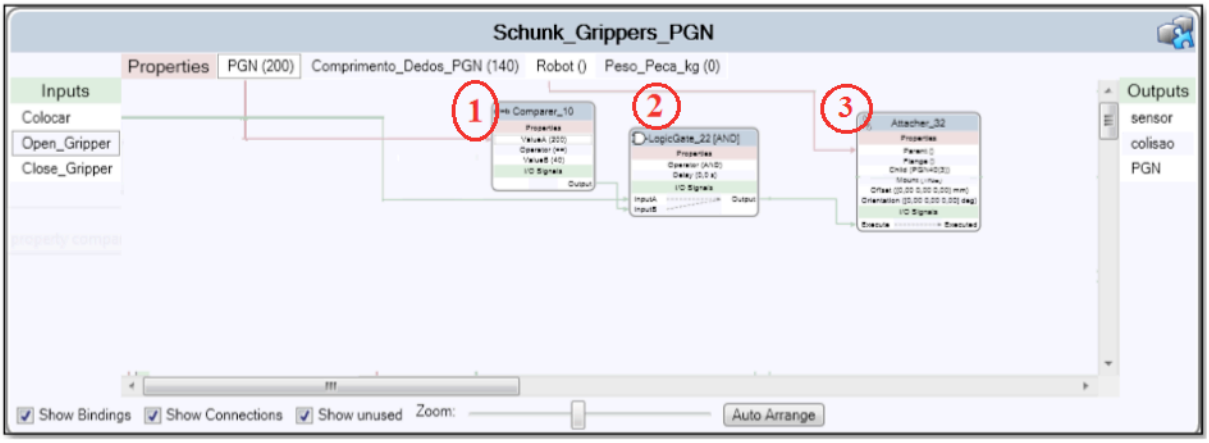


Figura 3.17 – Lógica implementada para a escolha da garra

Configuração do comprimento dos dedos da garra

O comprimento dos dedos é criado através do SC *LinearExtrusion* (1 e 2), que faz com que seja feita uma extrusão na face escolhida, mediante a projeção definida. Sendo o comprimento mínimo dos dedos 40 mm, é necessário usar uma expressão matemática que retire esse valor ao inserido. É realizada uma operação matemática com o SC *Expression* (4) para que o valor introduzido esteja conforme as unidades presentes na geometria da garra. Como é possível observar na figura 3.18, é necessário um SC *VectorConverter* (3) para transformar o valor do comprimento dos dedos na componente "Z" de um vetor para que possa ser conectado ao SC *LinearExtrusion.*

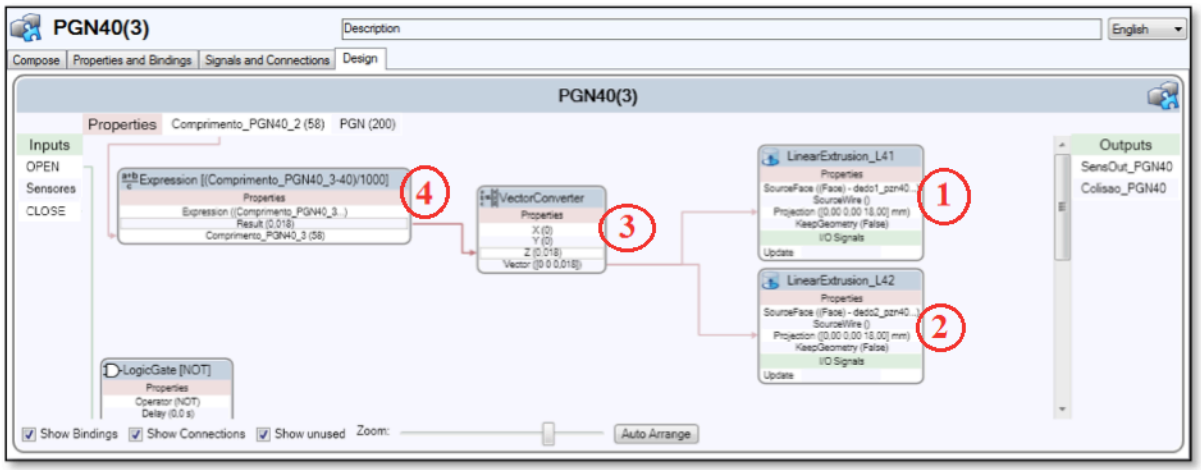


Figura 3.18 – Lógica para a extrusão dos dedos

Esta extrusão realizada na face superior dos dedos da garra é o que permite configurar o comprimento dos dedos. A alternativa seria a de serem modelados dedos de diferentes comprimentos e/ou geometrias e serem disponibilizadas essas alternativas ao utilizador. No entanto, com o SC *LinearExtrusion* é possível alargar as possibilidades no que toca ao comprimento dos dedos, ainda que se tenha que seguir uma configuração fixa.

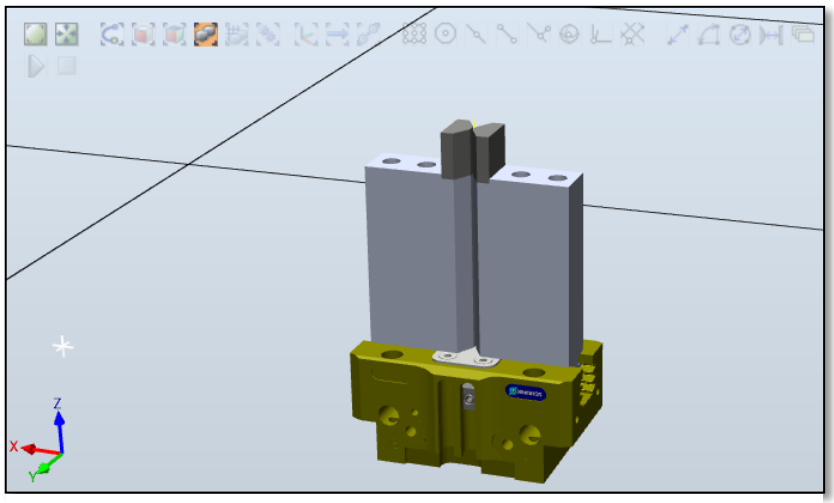


Figura 3.19 – Perfil dos dedos extrudidos

O SC *LinearExtrusion* cria uma geometria nova e para fazer com que esta faça parte do mecanismo garra, possuindo a mesma cinemática, é necessário que cada SC *LinearExtrusion* seja introduzido no *Link* correspondente ao dedo em que fez a extrusão (figura 3.20). O SC *Attacher* também presente no *Link* faz com que a parte criada fique agarrada ao dedo.

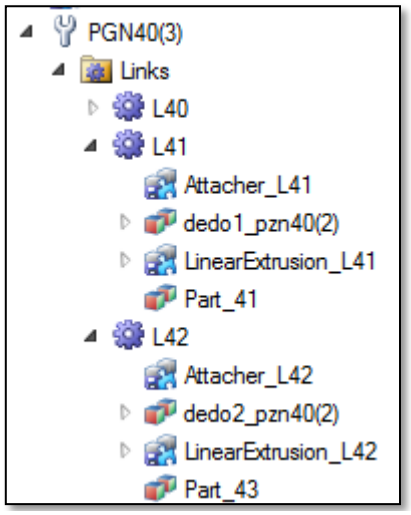


Figura 3.20 – SC dentro dos Links da garra

Como foi descrito nas tabela 3.1 (página 19) e 3.2 (página 20), cada garra possui um comprimento máximo permitido para os dedos. Para introduzir no SC a capacidade de verificar os limites do comprimento dos dedos, foi necessário criar uma lógica com recurso a dois SC *Comparer*: um SC *Comparer* (1) que é ativado quando o número de série da garra é o mesmo que está definido e o SC *Comparer* (2), que tem definido o valor do comprimento máximo dessa garra e que é ativado quando este é ultrapassado.

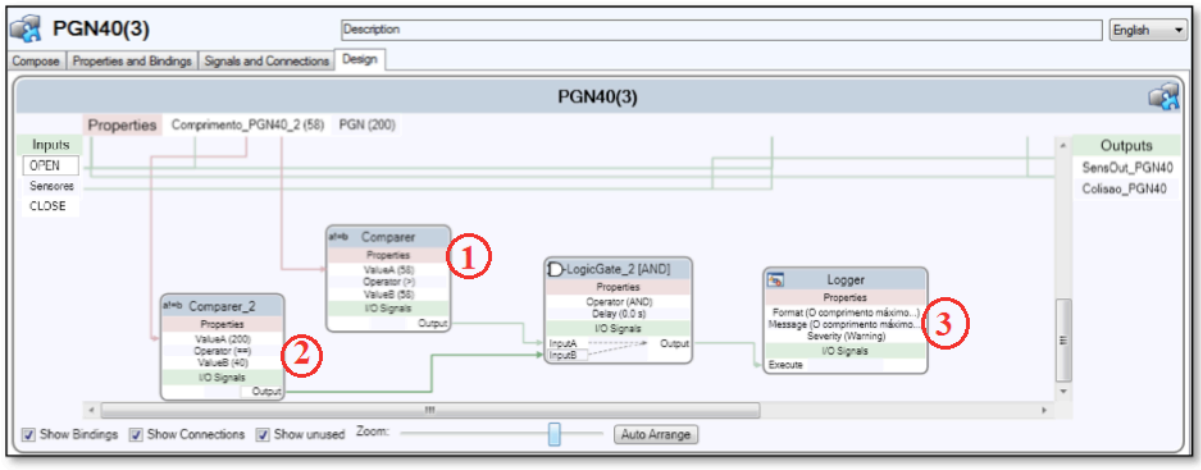


Figura 3.21 – Lógica para a verificação de erros de comprimento

A lógica anterior tem o objetivo de executar um SC *Logger* (3) quando o valor do comprimento ultrapassa o definido para a garra. Desta forma, na caixa de comandos do *software* aparecerá uma mensagem de aviso alertando para o facto de não ser possível que a garra escolhida tenha os dedos com o comprimento definido.

Na figura 3.22, é possível observar a mensagem de aviso apresentada na caixa de comandos, quando é escolhido um comprimento superior ao permitido. Esta mensagem tem um valor informativo, não impossibilitando ao utilizador a possibilidade de usar os dedos de forma errada.

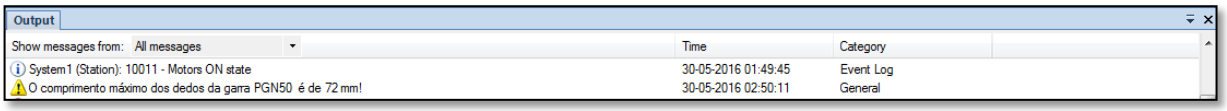


Figura 3.22 - Mensagem de erro de comprimento excessivo

Seleção do robô no qual a garra vai ser integrada

O objetivo após a escolha da garra a usar é a de a acoplar a um mecanismo que seja capaz de a manipular de forma a realizar os movimentos pretendidos. Assim, o SC dá a possibilidade ao utilizador de selecionar entre todos os mecanismos presentes na estação, qual o pretendido. Como é visível na interface do SC na figura 3.23, estão presentes quatro robôs, que são mecanismos aceites para fazer o acoplamento.

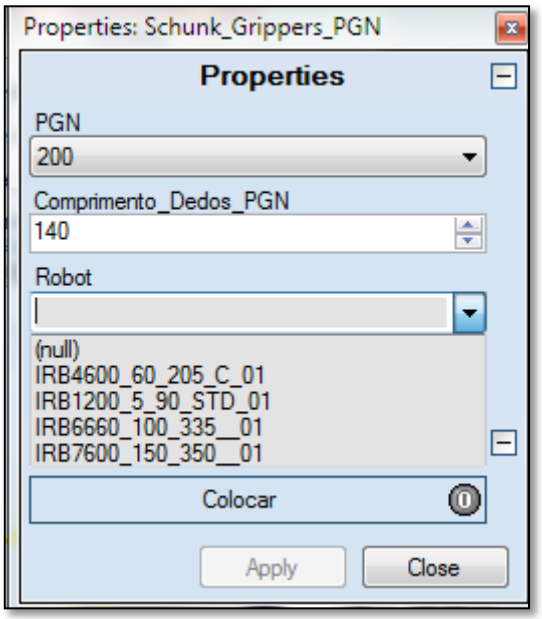


Figura 3.23 – Menu para a escolha do robô

Apesar de ser principalmente pensada para fazer a ligação a um robô, esta classe (*Mechanism*) permite ao utilizador acoplar uma garra a outro mecanismo que ele próprio tenha criado ou já existente no RobotStudio.

As garras criadas nestes SC não contemplam as ligações mecânicas que é necessário configurar para que as garras possam ser manipuladas pelos robôs/mecanismo, devido a cada um destes necessitar de uma ligação específica. No entanto, caso sejam criadas estas ligações e a garra seja colocada através deste SC, esta ficará acoplada corretamente.

Validação da adequação da garra para a manipulação da peça de determinada massa

Um dos parâmetros principais a definir para a escolha de uma garra é a capacidade da mesma de suportar a massa da peça. Cada garra tem uma massa máxima recomendada para as peças a manipular, portanto considerou-se benéfico acrescentar este parâmetro de forma a que o utilizador possa verificar se a garra que pretende acoplar no robô serve as suas pretensões ou ultrapassa a sua capacidade.

Esta funcionalidade é implementada através de uma lógica criada em cada SC de cada garra, que tem como propriedades o número de série da garra e a massa que é inserida no menu inicial de interface. O SC *And* (1) estabelece uma condicionante para quando for escolhida a garra onde se encontra este SC e a massa for superior ao permitido. Este valor é guardado em cada garra num SC *Comparer* (2) e é executado um SC *Logger* (3), que faz surgir um texto na caixa de comandos do programa. Esta lógica pode representada na figura 3.24.

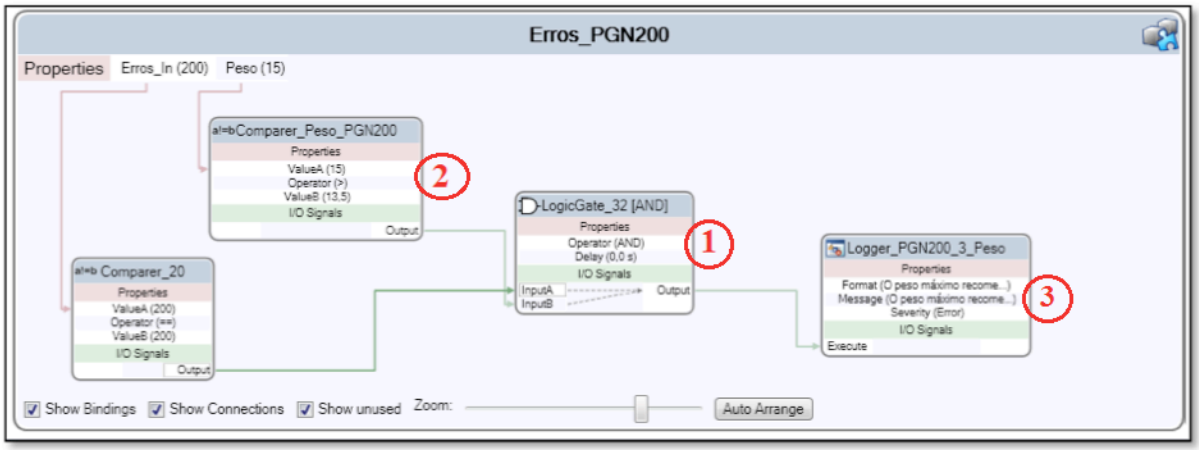


Figura 3.24 – Lógica para a verificação do peso recomendado

Esta mensagem (figura 3.25), está definida para ser visualizada como erro, não impedindo ainda assim a simulação e a geração da garra. A única massa que não pode ser ultrapassada é a de 30 kg, já que é o máximo definido pela variável nas propriedades do SC.

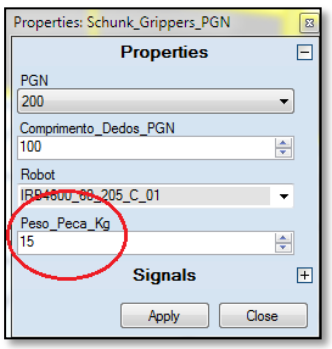


Figura 3.25 – Escolha do peso excessivo na interface do SC

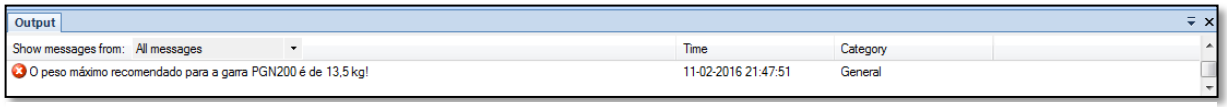


Figura 3.26 – Mensagem de erro de peso excessivo

Disponibilização de sensores de presença

Para a garra conseguir pegar no objeto é preciso seguir a lógica presente na figura 3.27. No volume de trabalho da garra é criado um SC *LineSensor* (1) com o objetivo de saber se a peça se encontra dentro desse volume para que possa ser agarrada. Deste SC sai um sinal digital quando este volume é invadido por uma geometria que permite, aquando da programação, a execução da ordem de fecho da garra. A peça presente no volume definido é ligada ao SC *Attacher*, o que faz com que esta fique agarrada depois de os dedos da garra terem colidido com a peça. A expressão matemática usada é para atualizar o comprimento do SC *LineSensor*, já que terá de estar relacionado com o comprimento dos dedos.

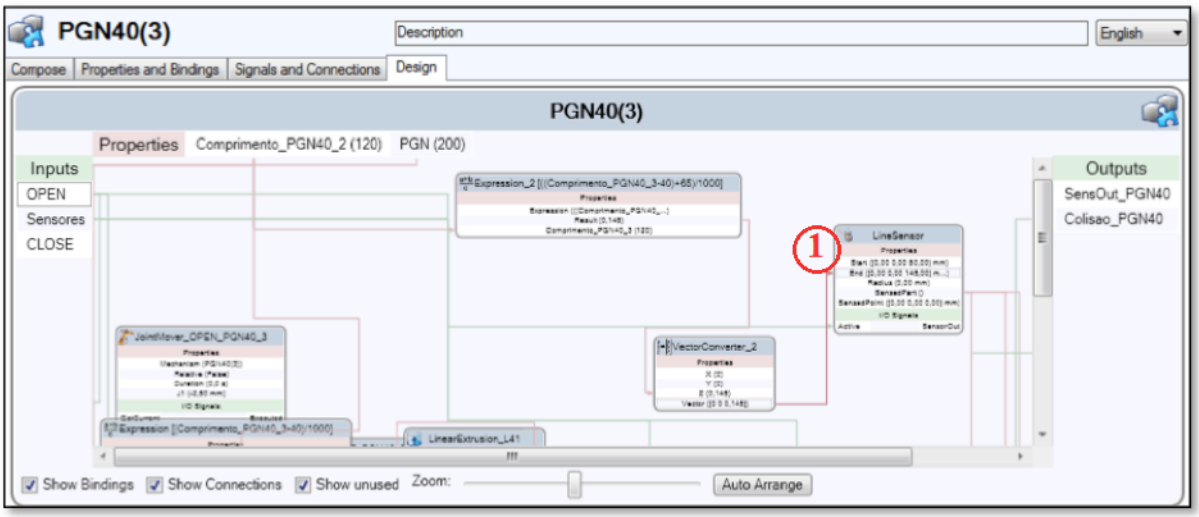


Figura 3.27 – Lógica implementada para o sensor de presença

Este sensor é bastante importante para a programação em linguagem RAPID aquando da integração dos SC na célula robótica. Não sendo uma representação de um sinal fisicamente implementado na garra (ainda que seja possível através dos vários acessórios da Schunk), este é necessário para no programa ser possível executar a ordem de fecho da garra somente quando a peça se encontra passível de ser agarrada, evitando assim possíveis erros no processo.

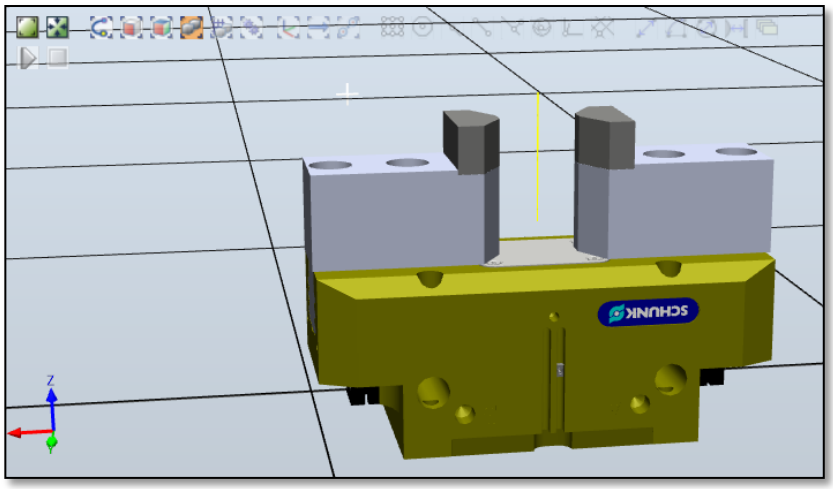


Figura 3.28 – Sensor de presença

Disponibilização de sensores de colisão (garra/peça)

Outro sensor criado com a finalidade de proporcionar uma simulação mais próxima da realidade é o sensor de colisão. A lógica é criada de forma a este estar ativo quando houver contacto entre os dedos e qualquer objeto presente na estação, como se pode observar na figura 3.29. O objeto sentido pelo SC *LineSensor* (1), que é a peça que se pretende agarrar, é ligado ao SC *CollisionSensor* (2) de forma a ser sobre esse objeto que é tida em conta a colisão. Quando a colisão é detectada, faz executar um SC *Attacher* (3), para que a o objeto se mova de forma solidária com a garra, simulando desta forma, no ambiente virtual, o comportamento real de agarrar uma peça. Da forma inversa, quando a garra é aberta, o SC *Detacher* (4) é executado, sendo que a peça que previamente havia sido presa é libertada deixando de se movimentar de forma solidária com a garra.

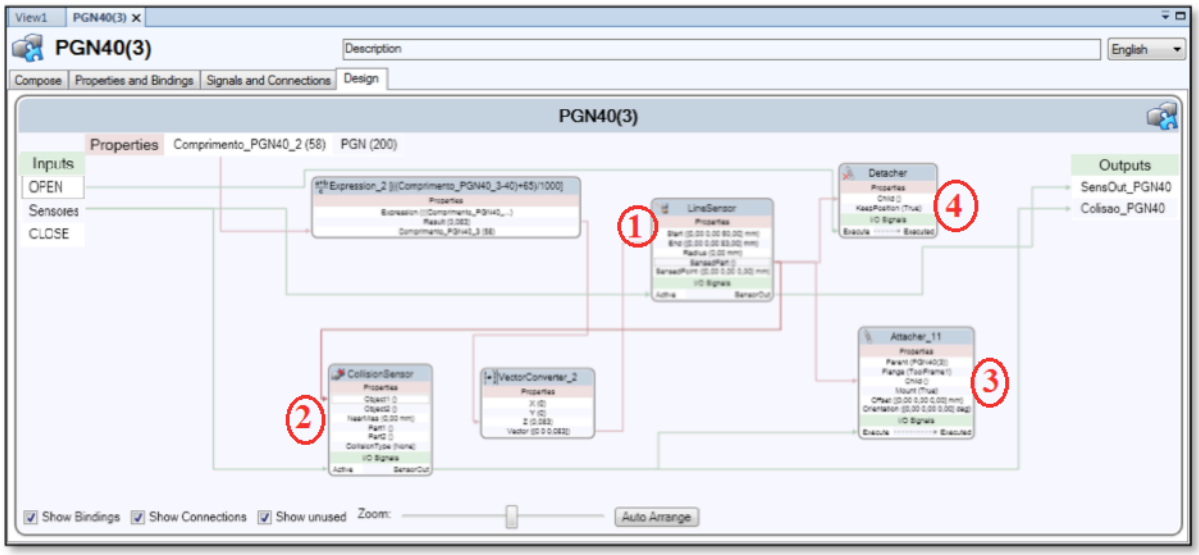


Figura 3.29 – Lógica implementada para o sensor de colisão

Tanto o sensor de presença como o sensor de colisão são sinais criados para que na programação possam ser usados para configurar eventos necessários. São por isso criados sinais de *Output* ligados a estes SC de forma a que possam ser lidos e controlados os seus valores em cada momento da simulação.

No caso do sensor de colisão, quando o SC *CollisionSensor* se encontra ativo é enviado um sinal para o *output* "Colisão_PGN40". Desta forma, durante a programação o sinal permite definir um evento, por exemplo esperar que os dedos da garra colidam com a peça para proceder à manipulação desta para outro ponto. Além disto, é também possível fazer uma monitorização dos sinais durante a simulação, através de uma consola que contém os sinais presentes no controlador, facilitando assim ao utilizador a deteção de erros.

O sinal de entrada "Sensores" é um sinal apenas de leitura, configurado para ter sempre o valor "1" e tem como finalidade fazer com que tanto o sensor de presença como o de colisão estejam sempre ativos. De outra forma não seria possível sentir mais do que uma peça durante uma simulação.

Sinais de abertura/fecho da garra

Para ser realizada a abertura e fecho das garras são criados dois sinais de *input* para simular o circuito pneumático previamente apresentado. Esta lógica é implementada através de um *SC Latch* (3), que permite fazer o *Set* e o *Reset* de outros dois SC. Como é possível observar na figura 3.26, o sinal "OPEN", faz o *Set* do SC *JointMover_Open* (1) que está programado para a abertura dos dedos do mecanismo associado. O sinal "CLOSE" faz o *Reset* do SC *Latch* (3) fazendo assim executar o SC *JointMover_Close* (2) que faz o fecho da garra, conforme se observa na figura 3.30.

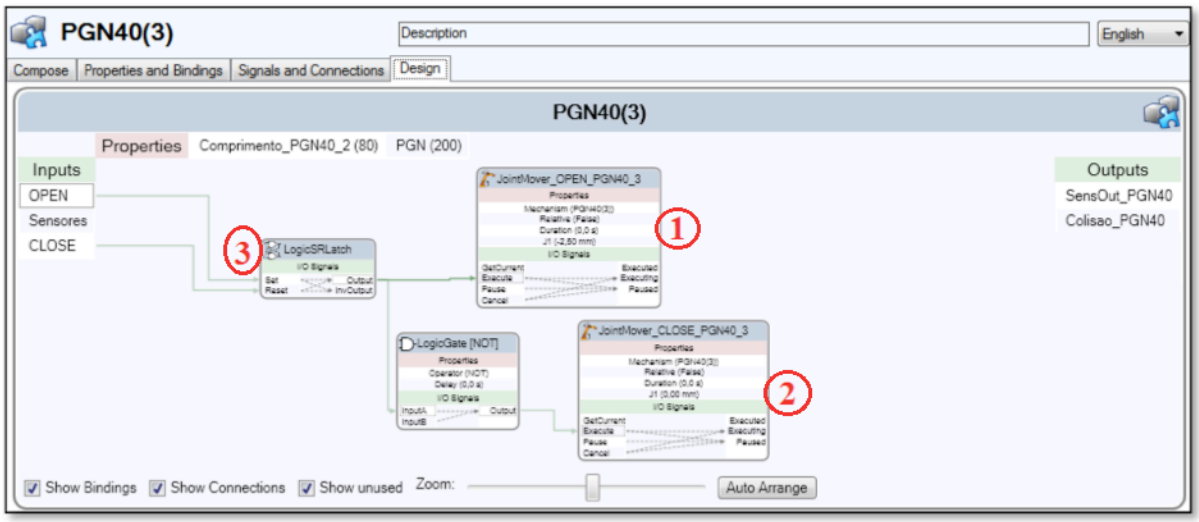


Figura 3.30 – Lógica implementada para a abertura/fecho da garra

O SC *JointMover_Close* (3) está também conectado ao sensor de colisão que, como foi descrito anteriormente, impede que visualmente haja uma sobreposição entre a peça e os dedos, momento que pode ser observado na figura 3.31. Quando é detectada a colisão, um sinal é enviado para fazer o *Pause* no movimento de fecho da garra. Só assim é possível que a garra adopte uma posição intermédia.

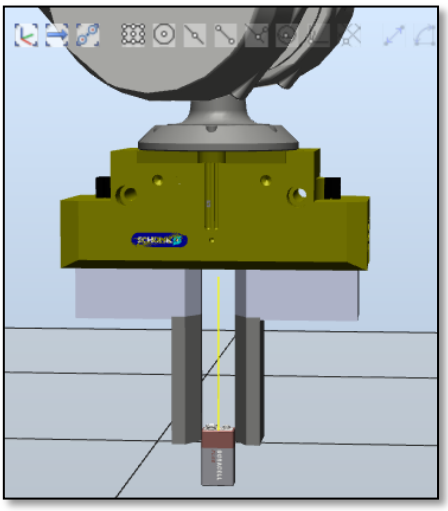


Figura 3.31 - Colisão da peça com os dedos da garra

3.3 Síntese do *Smart Component* criado

No SC criado para fazer a escolha da garra pretendida são criadas quatro propriedades dinâmicas como se observa na figura 3.32. A primeira é a que nos permite escolher a garra, sendo que é uma variável numérica, "Int32", e cujos valores atribuídos são os dez números de série das PGN. O comprimento dos dedos também é uma variável do tipo numérica, que se encontra limitada pelos valores mínimo e máximo definidos para os dedos da garra, 40 mm e 350 mm, respetivamente. A propriedade "Robot", pertence à classe *Mechanism*, e serve para o utilizador fazer o acoplamento da garra escolhida a um mecanismo pertencente à estação. Por último, a massa da peça "Peso_Peca_kg", também do tipo numérico, que também tem valores mínimos e máximos, de acordo com as capacidades das garras.

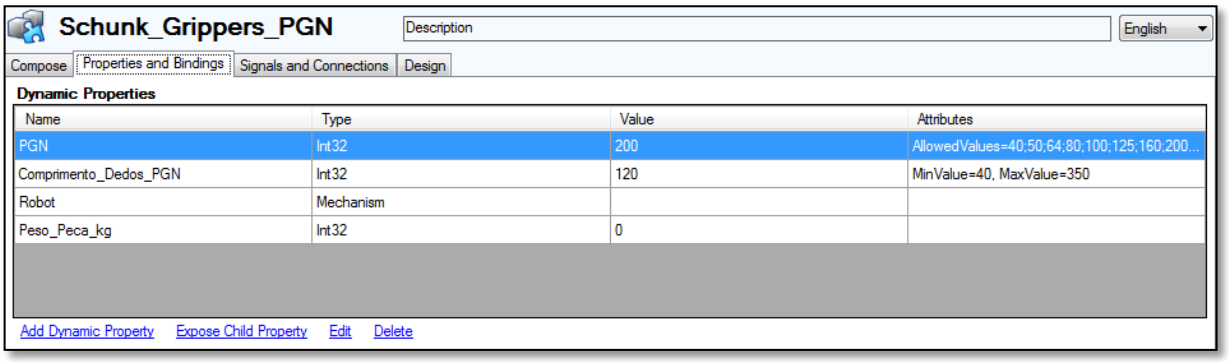


Figura 3.32 – Propriedades do SC Schunk_Grippers_PGN

Quanto à classe de sinais, é possível observar na figura 3.33 que foram criados seis sinais, três de entrada e três de saída. O sinal "Colocar" é uma entrada digital que executa o SC *Attacher* e dessa forma procede ao acoplamento escolhido pelo utilizador. Os restantes dois sinais de entrada são o "Open_Gripper" e o "Close_Gripper" que fazem a abertura e o fecho da garra. Os dois sinais digitais de saída são os sensores de presença da peça e de colisão com os dedos da garra, para que estes possam ser lidos pelo controlador e contribuir para a programação em linguagem RAPID. O outro sinal de *output* é analógico e pretende enviar para o controlador a informação sobre qual a garra escolhida pelo utilizador, de forma a utilizar a *Tooldata* respetiva.

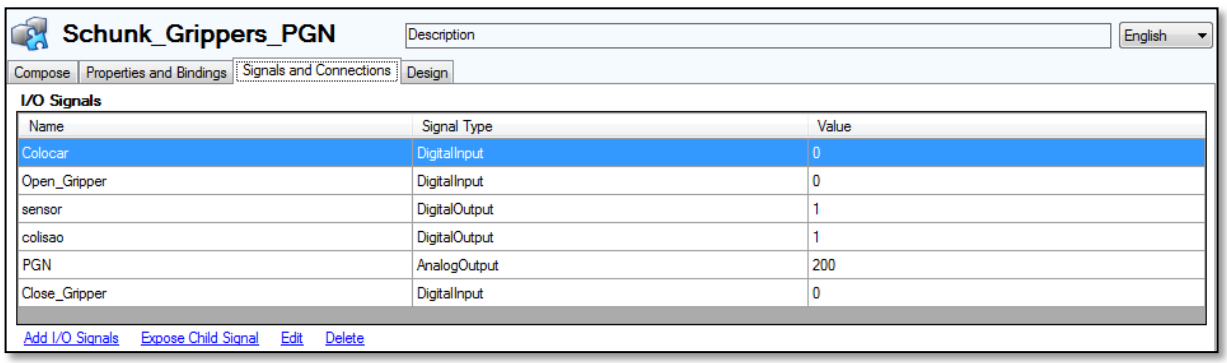


Figura 3.32 – Sinais do SC Schunk_Grippers_PGN

Quando é desenvolvido SC, na janela de *design* da lógica da estação é possível obter o seu bloco lógico que resume as suas propriedades e sinais. Nesta janela estas podem ser ligadas aos restantes componentes presentes na célula e construir os comportamentos lógicos pretendidos. Este bloco pode ser observado na figura 3.34.

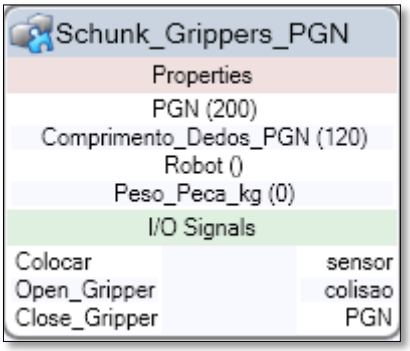


Figura 3.34 – Bloco SC Schunk_Grippers_PGN

Todos os SC das garras foram construídos com recurso à criação de duas propriedades, que são subpropriedades do SC "pai": a extrusão dos dedos que faz a configuração do comprimento e a escolha da garra. Para estas características poderem estar interligadas, o tipo de variável deve ser o mesmo, como se pode ver na figura 3.35.

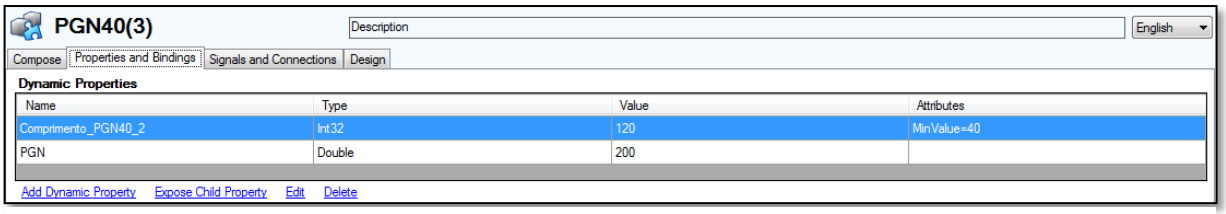


Figura 3.35 – Propriedades SC PGN/PZN

Cada garra tem cinco sinais associados, sendo que três são de entrada e dois de saída, como está explícito na figura 3.36. Estes sinais também estão de acordo com o SC "pai", já que o objetivo é a leitura destes sinais nesse SC, mas toda a lógica é implementada individualmente no SC de cada garra.

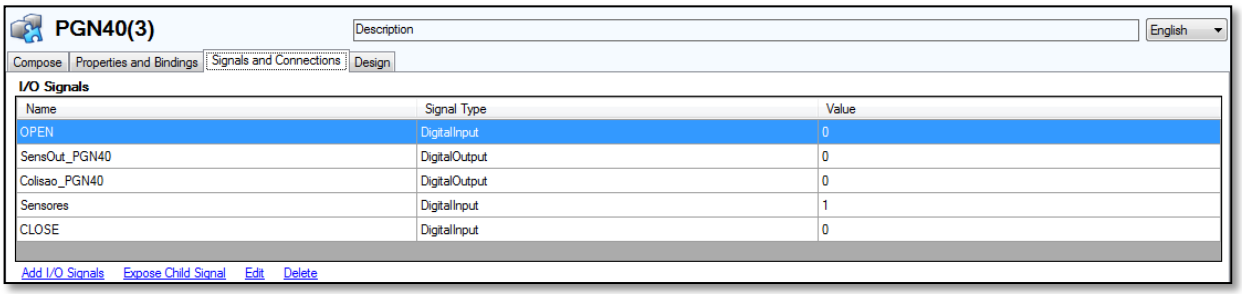


Figura 3.36 – Sinais SC PGN/PZN

Por fim, o bloco lógico correspondente a cada SC das garras, tem o aspecto que pode ser observado na figura 3.37.

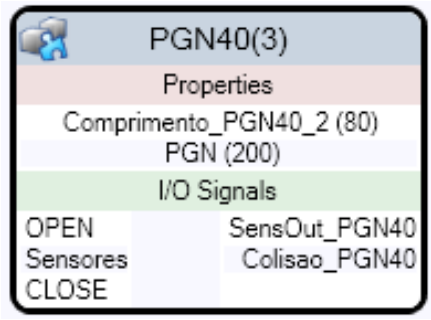


Figura 3.37 – Bloco do SC PGN/PZN

Estão concretizados então todos os passos seguidos para a criação destes SC. É possível observar na figura 3.38, a interface visual final que contém os quatro parâmetros que podem ser definidos pelo utilizador e o botão que faz com que a garra seja acoplada ao mecanismo escolhido.

Este menu é criado automaticamente pelo RobotStudio, após serem associadas ao SC as diferentes propriedades e sinais. Algumas dessas são definidas como variáveis apenas de leitura, não aparecendo assim visualmente apresentados neste menu. Doutra forma teríamos um conjunto de características sem possibilidade de parametrização misturadas com aquelas que se pretende serem parametrizáveis.

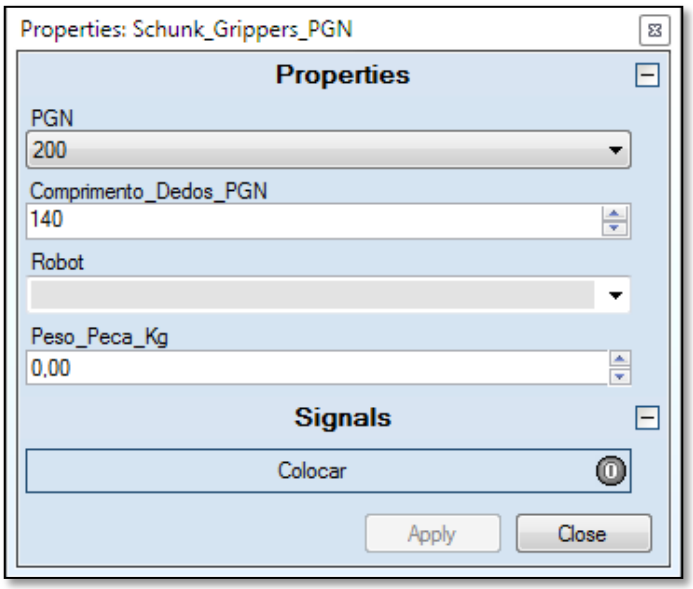


Figura 3.38 – Interface do SC Schunk_Grippers_PGN

3.4 Comunicação com o controlador

Além dos ficheiros dos SC que são criados, este projeto tinha como o objetivo estes serem testados numa estação e para tal é necessário fazer a ligação dos sinais criados ao controlador da ABB, para que possa ser configurado um programa através do RAPID.

O RobotStudio permite a criação de um tipo de ficheiro "CFG" que grava sinais criados no controlador e que podem assim ser carregados, facilitando a construção da lógica da estação e das variáveis a programar.

Foi então criado um ficheiro que contém os cinco sinais, três de entrada e dois de saída, presentes no SC quer das garras PGN como das PZN, como está representado na figura 3.39. Estes sinais ficam guardados no controlador e é com estes que são construídas as rotinas pretendidas.

T_ROB1/Module1 | I/O System

Name	Type	Value	Min Value	Max Value	Simulated	Network	Device	Device Mapping	Category	Label
colisao	DI	0	0	1	Yes	<none>	<none>			
Open_Gripper	DO	1	0	1	Yes	<none>	<none>			
PGN	AI	200	0	0	Yes	<none>	<none>			
sensor	DI	1	0	1	Yes	<none>	<none>			
Close_Gripper	DO	0	0	1	Yes	<none>	<none>			

Figura 3.39 – Sinais criados no controlador

Para além da criação das variáveis no controlador, é necessário que estas estejam conectadas às dos SC criados, o que tem de ser realizado manualmente. É possível observar na figura 3.40 que os nomes dados às variáveis criadas são iguais para que as ligações possam ser feitas de forma intuitiva.

Sendo que só é possível fazer ligações entre sinais e não entre propriedades, foi necessária a criação de um sinal PGN e PZN, que é um sinal analógico contendo o valor numérico da série da garra, para que na programação possa ser escolhido o TCP correto.

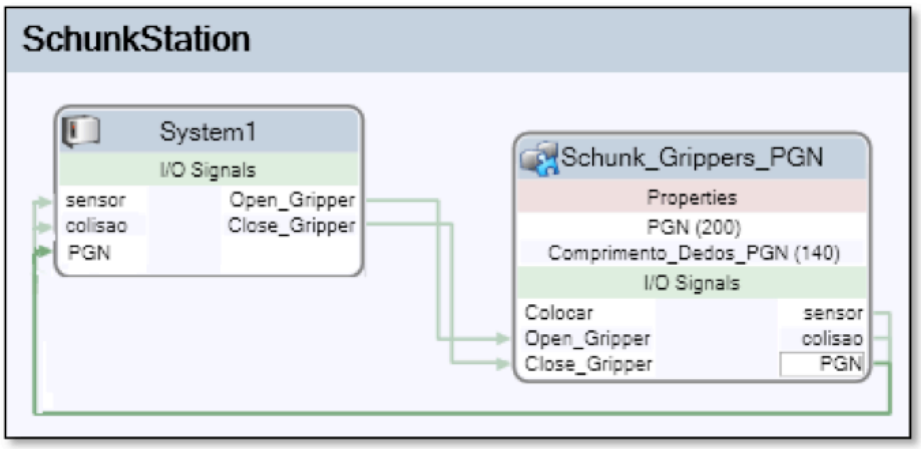


Figura 3.40 – Lógica de contactos na estação

3.5 Simulação

O comportamento da garra na simulação é uma questão abordada na construção do SC, já que o objetivo é que este recrie, com o maior detalhe possível, os eventos que serão implementados na célula real.

Quando são importados os SC criados estes ficam localizados na estação, como é possível observar na figura 3.41, sendo que as garras só são acopladas ao robô quando é ativado o sinal "Colocar". No entanto, se não se pretender usar esse menu de interface, as garras podem ser retiradas individualmente do SC "pai", colocadas manualmente na posição desejada e a configuração da lógica instalada pode ser editada.

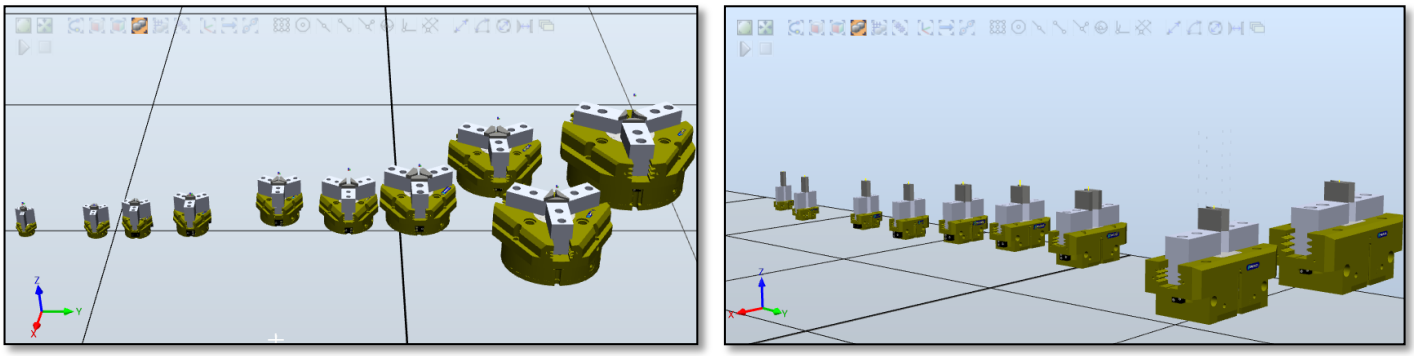


Figura 3.41 – Conjunto das garras PZN (à esquerda) e das PGN (à direita)

Na figura 3.42 está exemplificado o momento em que é dada a ordem de fecho a uma garra PGN e os dedos desta colidem com a peça. Como foi visto anteriormente, o movimento do fecho é parado no momento de colisão para que visualmente não haja sobreposição de geometrias.

Após a colisão entre os dedos da garra e a peça, estes mover-se-ão solidários um com o outro, até à posição programada. A peça só é libertada quando é feita a abertura da garra.

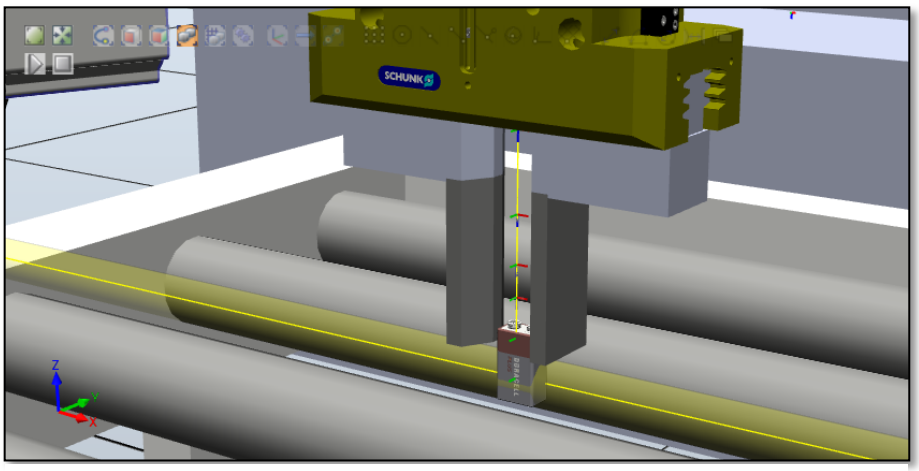


Figure 3.42 – Garra PGN em simulação

4 Integração de *Smart Components* em célula robótica

Neste capítulo é apresentada uma célula robótica que foi criada com a integração e programação dos SC criados anteriormente.

Com o objetivo de justificar, apresentar e verificar a utilidade dos SC criados, foi concebida uma célula robótica em que estes são usados. Desta célula robótica fazem parte uma série de componentes geométricos, visíveis na figura 4.1. Os SC criados foram importados para esta célula e foram ligados ao controlador virtual de forma a serem programados.

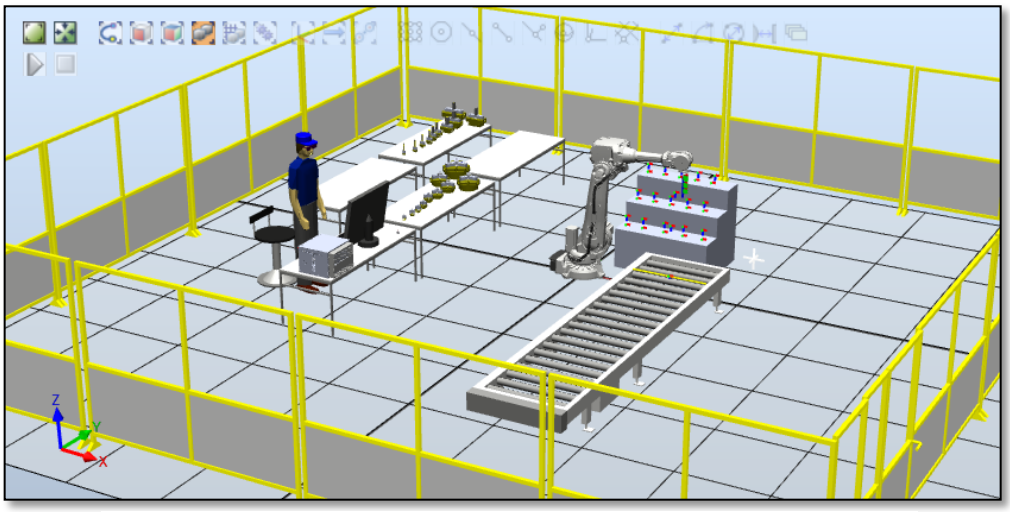


Figura 4.1- Célula conceptual criada

A célula tem como finalidade permitir um comportamento básico de manipulação de peças que são transportadas por um *conveyor* e que, com auxílio dos SC das garras que estão acoplados a um robô ABB, são colocadas numa posição pré-definida.

Nesta célula foi desenvolvido um SC adicional para que possa ser feito o teste às garras de uma forma mais rápida e eficiente. Os atributos deste SC também foram ligados ao controlador, aumentando assim as possibilidades de programação existentes.

A criação de um SC para realizar uma importação e configuração de um modelo geométrico no *software* era tido como um processo bastante expedito. Neste caso, a ligação dos componentes ao controlador e a sua programação seria o maior desafio e, daí, a necessidade da sua verificação. Neste capítulo são apresentados os passos realizados para tal.

4.1 Controlador IRC5

O IRC5 é um controlador de quinta geração da ABB, que serve para controlar todos os robôs listados na biblioteca do RobotStudio. A programação dos robôs é configurada por um computador, mas é tipicamente descarregada para um controlador através de um protocolo de comunicação, permitindo que o programa fique guardado no controlador e possa ser acedido sempre que necessário.

Uma das grandes potencialidades do software da ABB é que permite utilizar um controlador virtual fazendo com que seja possível desenvolver um programa de rotinas para o robô e simular o seu comportamento. Quando é importado qualquer robô da biblioteca, o software dá a possibilidade de importar também o controlador, de forma a que possa ser feita a programação do robô. Além de podermos importar o controlador virtual para a estação, também é possível importar o próprio modelo 3D do mesmo, como ilustrado na figura 4.2.

Figura 4.2 – Controlador IRC5 (modelo compacto)

Todos os sinais criados anteriormente nos SC pressupõem a sua utilização aquando da programação do robô. Para isso têm de ser criados neste controlador virtual os sinais de entrada e saída para serem interligados com os sinais de I/O dos SC. É assim possível utilizar esta interação para recriar os movimentos e as simulações de utilização dos SC.

4.2 *Smart Component* da célula

Para a construção da célula robótica, foi criado outro SC para que o teste às garras pudesse ser feito de uma forma mais expedita. O menu de interface deste SC pode ser visto na figura 4.3. Neste SC pode ser escolhido um de seis *conveyors* disponíveis, sendo que existem três valores pré-definidos para o comprimento (950, 1200, 1400) e dois para a largura (4000, 4600).

Em termos de escolha das peças a serem usadas para testar os SC, existem três possibilidades de geometrias. Pode ser criada uma caixa paralelepipédica, com a introdução

das 3 medidas associadas aos eixos "X", "Y" e "Z" da estação, um cilindro com a introdução do raio e altura ou uma peça personalizada, cuja geometria pode ser importada de qualquer software CAD, desde que seja num formato suportado pelo software. São definidas pelo utilizador as medidas das peças a criar para que as trajetórias do robô possam ser criadas de acordo com elas. Por último, é possível escolher a quantidade de objetos a serem colocados durante a simulação, sendo que os valores permitidos são 5, 10 ou 15. Estes valores correspondem a pretender o preenchimento de uma, duas ou as três prateleiras presentes na célula.

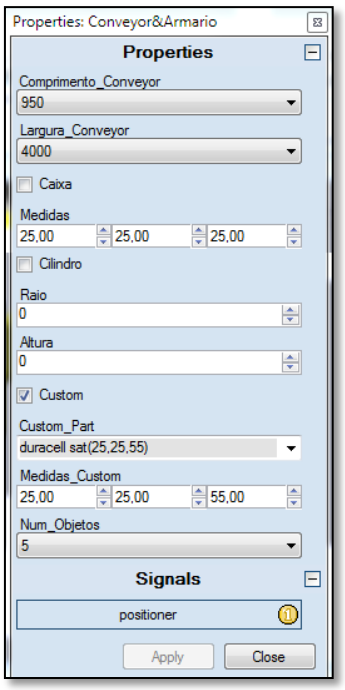


Figura 4.3 – Interface visual do SC da célula

Além dos parâmetros de entrada definidos pelo utilizador, para a programação em RAPID foram definidos cinco sinais de saída e um de entrada, como pode ser observado no esquema do SC *Conveyor&Armario* na figura 4.4.

Os sinais de *output* das medidas da peça escolhida, "Valor_X", "Valor_Y" e "Valor_Z", são convertidos em variáveis analógicas através de um SC e ligados ao controlador de forma a poderem ser lidos na programação. Assim é possível instruir o robô a fazer o seu movimento para que a peça se encontre dentro dos limites do SC *LineSensor* definido na garra.

O sinal "sensor2" está conectado a um SC *LineSensor* localizado no final dos *conveyors*. Quando é detectada a presença de alguma peça no espaço definido, faz comutar o seu sinal para "1" e assim parar o movimento do objeto. É este sinal que também faz com que seja dada a instrução ao robô para posicionar a garra, para proceder ao fecho e pegar na peça.

O *output* "Objetos" define o número de objetos que serão colocados nos pontos escolhidos. Este valor tem também de ser convertido numa variável analógica para que possa ser lido pelo controlador.

Com a lógica de funcionamento proposta na figura 4.4 é possível dispor de uma estação que permite testar inúmeras soluções de objetos para serem manipuladas com as garras criadas.

Os sinais criados são apenas uma proposta para que possam ser configurados os comandos para esta célula. No entanto eles próprios podem ser utilizados para diferentes configurações ou serem integrados numa estação com outros sinais de entrada e saída de modo a permitir construir comportamentos distintos dos implementados.

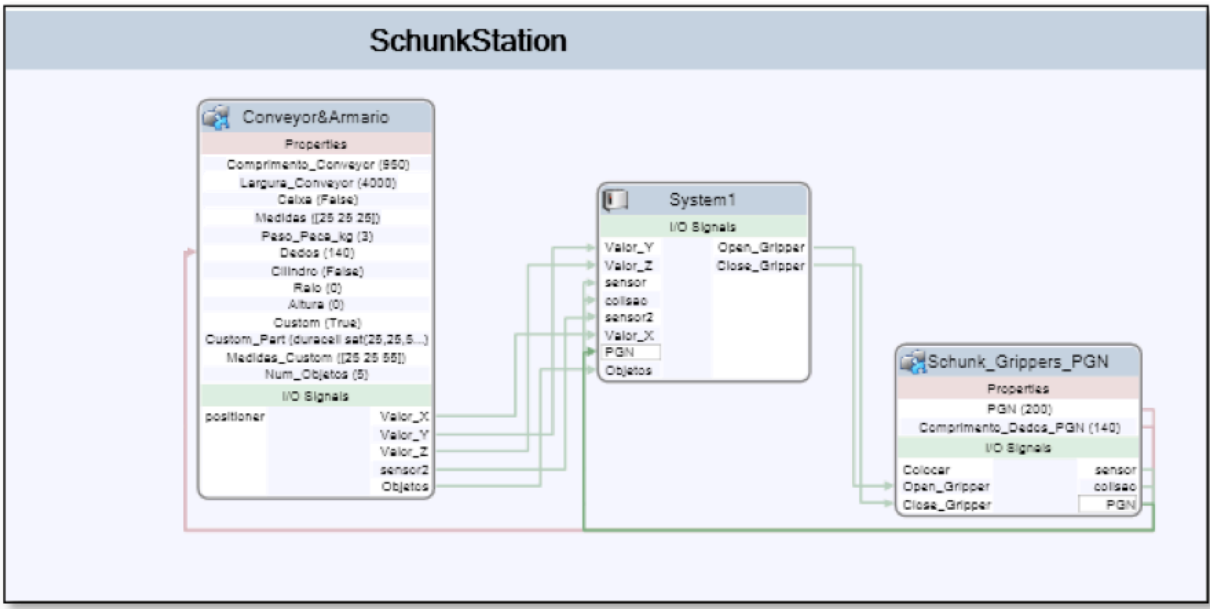


Figura 4.4 – Lógica da estação

4.3 Programação em RAPID

A linguagem RAPID é a que está presente no controlador dos robôs ABB. Esta linguagem permite especificar o tipo de movimento do robô, assim como controlar a lógica de funcionamento da estação, através de um conjunto de instruções. É possível ativar *Outputs* do controlador, ler o estado dos *Inputs*, definir o fluxo do programa, etc [15].

As instruções da linguagem RAPID podem ser agrupadas nas seguintes categorias:

- Instruções para controlo de movimento e posicionamento (relativas a tipo de trajetórias, aproximações a pontos de passagem, velocidades);
- Instruções para lidar com saídas e entradas (relativas ao manuseamento das entradas/saídas do controlador, utilizadas para a interligação com outros equipamentos e incluindo também temporizações);
- Instruções para controlo do programa (relativas a iniciar e parar programa, a instruções de salto, de ciclo, contagem de eventos e interrupções);
- Instruções de cálculo;
- Instruções de comunicação (para comunicação de dados do controlador com a consola de programação ou com um computador);
- Outras instruções (para a definição de parâmetros do sistema, mensagens de erro);

Dentro destes conjuntos de instruções, as que foram mais usadas foram as de controlo de movimento e as de interação com entradas e saídas.

Cada instrução de movimento contém (ver figura 4.5):

- Tipos de percurso (interpolação linear, circular ou de juntas) (a);
- Posição de destino (b);
- Velocidade (c)
- Dimensão de zona (d);
- Ferramenta a considerar (TCP ativo) (e).

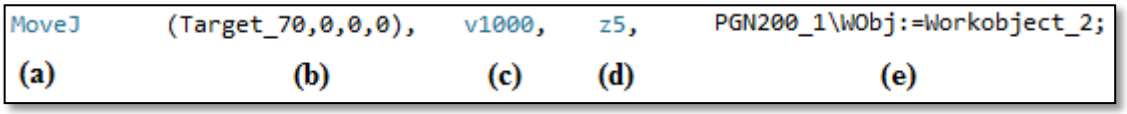


Figura 4.5 – Instrução de movimento

O robô e o controlador podem estar equipados com um determinado número de sinais digitais ou analógicos que podem ser modificados a partir do programa. O nome dos sinais é definido nos parâmetros do sistema e podem também ser lidos a partir do programa. O valor de um sinal analógico ou de um grupo de sinais digitais é especificado por um valor numérico.

É possível observar na figura 4.6, dois comandos usados com os sinais criados no controlador, o primeiro que faz com que o programa só avance quando o sinal "sensor" tome o valor "1", e o segundo provoca abertura da garra, já que o sinal é comutado para 1.

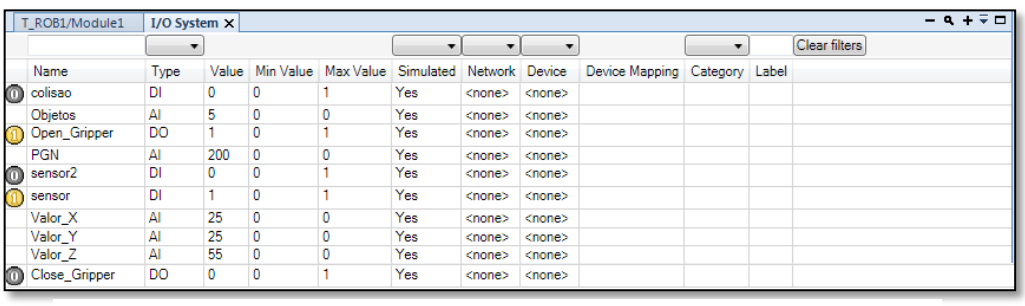
```
WaitDI sensor,1;

Setdo Open_Gripper, 1;
```

Figura 4.6 – Instruções de *Set*

Os sinais criados tanto nos SC como no controlador virtual têm como objetivo final a integração no desenvolvimento dos programas em linguagem RAPID tornando possível a configuração das rotinas de movimento e dos eventos da estação.

Pode ver-se na figura 4.7 o conjunto de sinais que foram definidos no controlador. Este conjunto apresenta sinais de diferentes tipos, analógicos e digitais.


T_ROB1/Module1 | I/O System

Clear filters

Name	Type	Value	Min Value	Max Value	Simulated	Network	Device	Device Mapping	Category	Label
colisao	DI	0	0	1	Yes	<none>	<none>			
Objetos	AI	5	0	0	Yes	<none>	<none>			
Open_Gripper	DO	1	0	1	Yes	<none>	<none>			
PGN	AI	200	0	0	Yes	<none>	<none>			
sensor2	DI	0	0	1	Yes	<none>	<none>			
sensor	DI	1	0	1	Yes	<none>	<none>			
Valor_X	AI	25	0	0	Yes	<none>	<none>			
Valor_Y	AI	25	0	0	Yes	<none>	<none>			
Valor_Z	AI	55	0	0	Yes	<none>	<none>			
Close_Gripper	DO	0	0	1	Yes	<none>	<none>			

Figura 4.7 –Sinais criados no controlador

Na figura 4.8 é apresentada a definição dos TCP (*Tool Center Point*) das garras PGN, informação que é armazenada no programa. Quando é escolhida uma determinada garra, a variável analógica PGN permite a escolha do TCP correspondente.

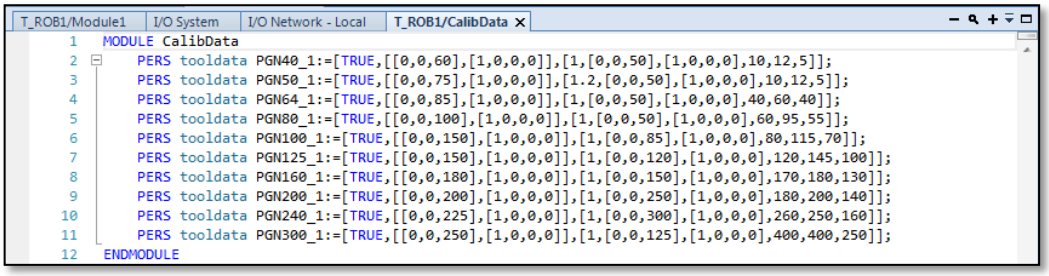


Figura 4.8– TCP das garras PGN

As coordenadas dos pontos criados (*Targets*) na estação também são guardados no ficheiro, para que possam ser utilizados na configuração da estação.

O programa foi desenvolvido num único módulo, com uma lógica de funcionamento construída de forma condicional. Foram escritos vários ciclos de movimentações das garras, que simulam a estação de *Pick and Place* para os pontos definidos, mas cada um desses ciclos é efetuado com a *tooldata* respetiva. Uma condicionante inicial "If" compõe cada um desses ciclos, para a seleção da *tooldata* e do número total de objetos a posicionar. Na figura 4.9 é apresentado um excerto do programa definido.

Figura 4.9 – Programa em RAPID

Foram criados pontos chave para a movimentação do robô, mas dada a possibilidade do utilizador escolher objetos de diferentes dimensões, seria impossível definir um único ponto para que a garra pudesse agarrar o objeto. Ultrapassou-se o problema criando três variáveis analógicas de entrada que contêm o valor das dimensões da peça e esses valores são usados na escrita do código para fazer o ajuste individual dos pontos para os quais é movimentada a garra. É possível observar na figura 4.8 que esta capacidade é criada através do comando *RelTool,* que faz uma compensação à posição definida do ponto (*Target*), com os valores que foram introduzidos pelo utilizador e gravados no controlador.

```
MoveL RelTool (Target_10,((AInput(Valor_X)/2)),((AInput(Valor_Y)/2)),(-(AInput(Valor_Z))+10)),
```

Figura 4.10 - Ajuste das medidas

4.4 Simulação da célula

Para esta célula foi escolhido um robô de estrutura revoluta e de seis eixos (IRB 4600-60/2.05), com uma capacidade de carga de 60 kg e um alcance de 2.05m [16], que está representado na figura 4.11. Este robô foi escolhido por ter uma capacidade de carga sempre superior à soma do peso da maior garra e das peças usadas, e também pelo respetivo alcance permitir movimentação até aos pontos definidos.

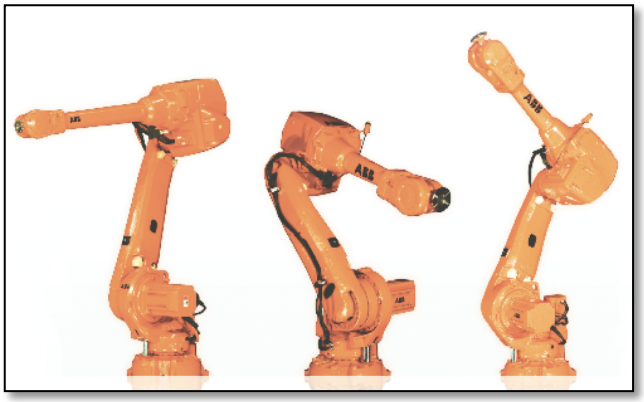


Figura 4.11 – Robô IRB4600-60/2.05 da ABB

Fazendo correr a simulação da célula, o utilizador pode escolher um conjunto de sinais cujos valores quer observar enquanto esta está a decorrer (figura 4.12). Neste caso foram escolhidos somente os sinais criados no controlador para esta célula (ex. "Valor_X", "Open_Gripper", "sensor2", etc.).

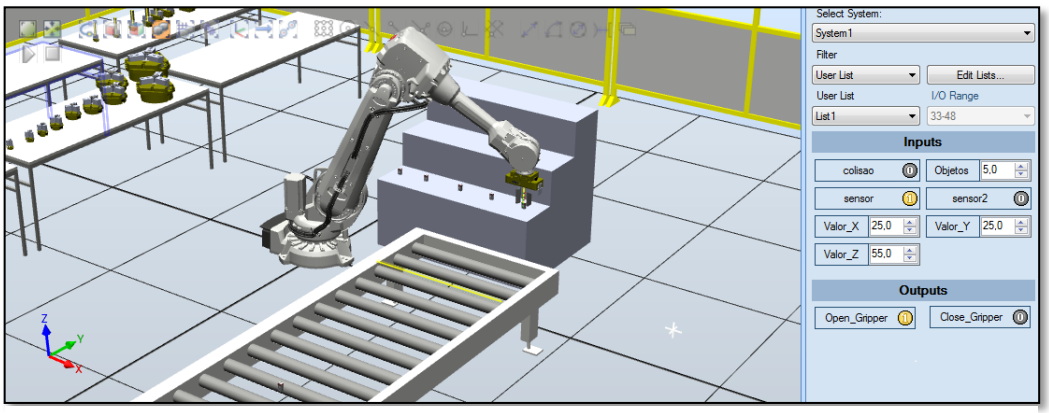


Figura 4.12 – Célula em simulação

A construção desta célula incorporando os SC criados permitiu validar as funcionalidades implementadas em cada SC, em particular a ligação destes ao controlador e a configuração geométrica das garras.

A facilidade de configuração, integração, e utilização destes SC pode ser útil para testar alterações a implementar no comportamento da célula criada e assim aumentar a sua flexibilidade. Por exemplo, é possível acomodar mais facilmente alterações aos produtos a manipular, dentro dos limites dos SC das garras existentes.

As funcionalidades incorporadas no SC revelaram-se ainda úteis na validação da simulação executada, através da monitorização de colisões da garra com outros objetos; no reconhecimento de presença do objeto a manipular e no estado de atuação da garra.

5 Conclusões e trabalhos futuros

Para facilitar a criação de uma célula robótica em que seja necessária a incorporação de ferramentas de manipulação foram modeladas duas séries de garras da Schunk. Além da criação geométrica das garras, um comportamento lógico foi implementado nas mesmas, com a finalidade de estas poderem ser programadas no próprio software. De forma a simplificar a utilização das séries, foi também criado uma interface que possibilita não só a escolha da garra pretendida, como a configuração/validação do comprimento dos dedos; o robô/mecanismo no qual a garra escolhida será acoplada e a validação da capacidade da carga da garra. Por fim, foi criada uma célula robótica que pretende ilustrar a forma como o componente desenvolvido pode ser integrado e, de facto, proporcionar ao utilizador uma vantagem para a seleção destas garras. Para que esta célula permitisse uma verificação mais expedita dos SC criados, foi desenvolvido outro SC. Nesta célula foram feitas as ligações necessárias entre os SC e o controlador para que fosse possível a programação de um robô.

Apesar dos SC das garras representarem apenas duas séries, o procedimento apresentado pode ser replicado de forma a abranger qualquer tipo de garra. Pode também ser usado como ponto de partida para a construção de comportamentos mais complexos seja com estas garras ou com outros componentes que também sejam usados em células robóticas.

O RobotStudio revelou assim possuir uma capacidade vasta na criação destes componentes inteligentes. A interligação entre os SC de base proporciona inúmeras possibilidades de implementação de comportamentos lógicos, podendo por isso ser utilizada para a construção de ferramentas que possam ser, de alguma forma, parametrizáveis, agilizando assim o processo de criação de uma célula robótica. A principal contrariedade verificada na criação dos SC dentro do ambiente do RobotStudio é a de ser complicada a utilização da janela de design quando é necessário recriar comportamentos complexos, com recurso a um número elevado de SC de base. A multiplicação dos comportamentos terá de ser realizada sem acesso aos blocos lógicos. O desenvolvimento deste tipo de comportamentos mais complexos pode ser realizado de uma forma mais expedita no caso dos SC serem criados através de código C#.

A interface dos SC desenvolvidos é *user-friendly* e bastante intuitiva, permitindo assim a sua utilização a utilizadores menos familiarizados com o *software.*

Apesar da importação destes SC para o *software* não ser problemática, foi criado em anexo um tutorial de como o fazer, mas tendo como principal objetivo apresentar as ligações necessárias a fazer na lógica da estação de modo a poder usar os sinais e propriedades criados na programação em RAPID. Isto torna ainda mais expedito o processo de criação de uma estação já que desta forma as variáveis são guardadas no controlador, sendo assim somente necessário o desenvolvimento do programa.

Como trabalhos futuros, seria interessante a integração, tanto dos SC das garras, como do que foi criado para o auxílio à célula e alguns componentes geométricos que estão normalmente presentes nestas células (barreiras, mesas, etc.), num *Add-in* em que a sua utilização fosse ainda mais simples.

Uma limitação dos SC que foram desenvolvidos diz respeito à configuração geométrica escolhida para os dedos da garra. Esta escolha deve-se à limitação existente no processo de configuração de SC quando programado a partir do RobotStudio. Na indústria são utilizadas garras cujos dedos possuem as mais variadas formas, partindo dos dedos que são fornecidos pelas marcas para posteriormente serem maquinados com a geometria adaptada à sua funcionalidade. A introdução da funcionalidade de poder importar a geometria de cada dedo que o utilizador pretendesse para a garra seria também uma funcionalidade que resolveria esta limitação.

Os SC desenvolvidos nesta dissertação foram criados com recurso à integração de outros SC de base já presentes no RobotStudio. Sendo que os SC também podem ser criados através de programação em C#, seria interessante para um trabalho futuro, não só comparar a facilidade de implementação de uma lógica semelhante à deste caso, como a exploração das capacidades e limitações da criação dos SC através de programação.

Referências

1. O'Neill, James, (1944), "*Prodigal Genious: The Life of Nicola Tesla"*
2. [Online] Available: http://kellenj.weebly.com/uploads/2/3/9/9/23998921/7590487_orig.gif, Junho 2016
3. **International Federation of Robotics**. [Online] Availabe: http://www.expo21xx.com/news/ifr-report-industrial-robots-market/ (2016)
4. **ABB**, WebPage. [Online] Available: http://new.abb.com/products/robotics
5. **ABB**. Operation Manual: RobotStudio 6.01. Document ID 3HAC032104-001 Revision P. 2015
6. Ramos, Richard. Schunk Gripper. **ABB RobotApps**. [Online], Available: https://robotapps.robotstudio.com/Details.aspx?fileId=49a97f61-3819-4a95-8a7a-13058076167a
7. **ABB**. Software Development Kit, [Online], Available: http://developercenter.robotstudio.com/DevCenter.aspx?DevCenter=RobotCommunication_PCSDK
8. **ABB**, Walkthroughs, Smart Components, [Online] Available: http://developercenter.robotstudio.com/Index.aspx?DevCenter=RobotStudio&OpenDocument&Url=html/6825b20c-b218-4bb5-896c-11f4e62ffaba.htmABB, RobotApps. Online Available: https://robotapps.robotstudio.com/List.aspx?categoryid=c6c12cc4-137e-4c7e-a3ea-79086dc1001d&category=Smart%20Component
9. **ABB**, RobotApps. [Online] Available: https://robotapps.robotstudio.com/List.aspx?categoryid=c6c12cc4-137e-4c7e-a3ea-79086dc1001d&category=Smart%20Component
10. Kong, Hope. Parametric Fence. **ABB RobotApps**. [Online] Available: https://robotapps.robotstudio.com/Details.aspx?fileId=5cddd811-319f-488a-938f-d3defec3db00
11. Kopzack, Joseph. "Rapid Development of robotic concept stations in ABB RobotStudio", 2013
12. **Schunk**, The Schunk PGN-PLUS Gripper, [Online], Available: https://es.schunk.com/fileadmin/pim/docs/IM0017880.PDF
13. **Schunk**, 2-Finger Parallel Gripper PGN-plus 40 - 380 Assembly and Operating Manual, Edição 03.05, 2014, Online, Available: http://gb.schunk.com/fileadmin/pim/docs/IM0004096.PDF
14. **Schunk**, 3-Finger Centric Gripper PZN-plus Assembly and Operating Manual, Edição 04.01, 2015, [Online], Available: http://gb.schunk.com/fileadmin/pim/docs/IM0004120.PDF
15. P.Abreu, Manual de utilização RobotStudio – Parte 2 – Programação, Edição 1.3, Porto: FEUP, 2012
16. **ABB**, ABB [Online]. Available: https://library.e.abb.com/public/eadd388d8bec4b75a3342026c1631b5e/IRB%204600%20ROB0109EN_H.pdf

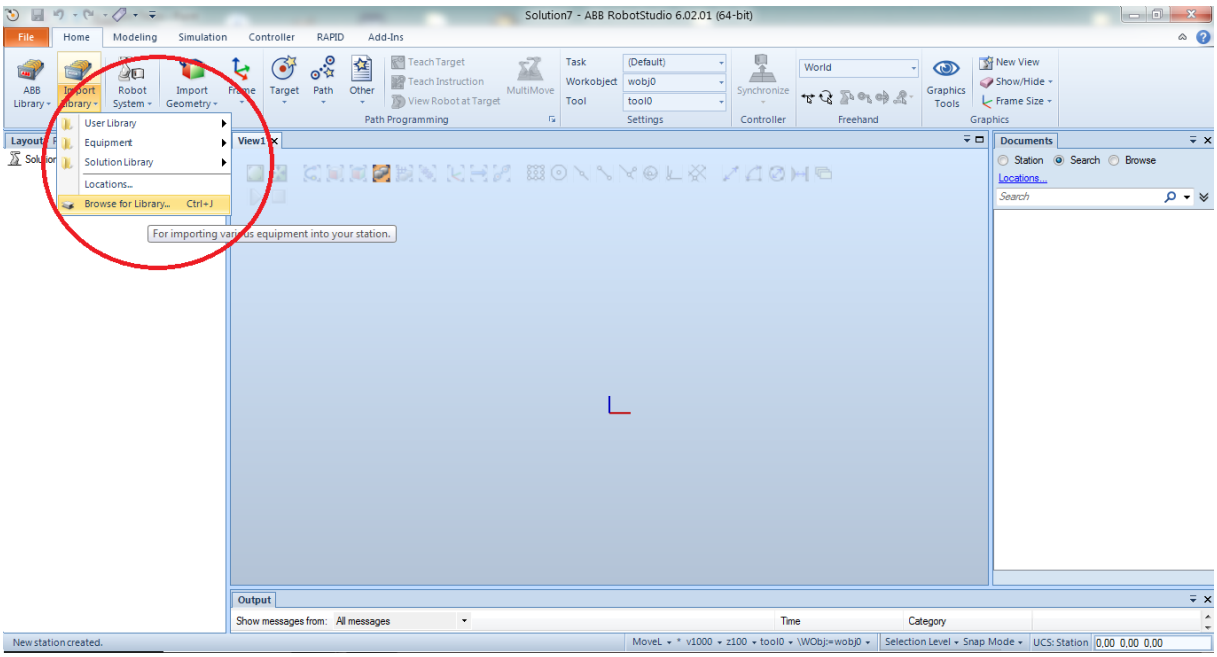
ANEXO A:

Manual de utilização
Smart Components

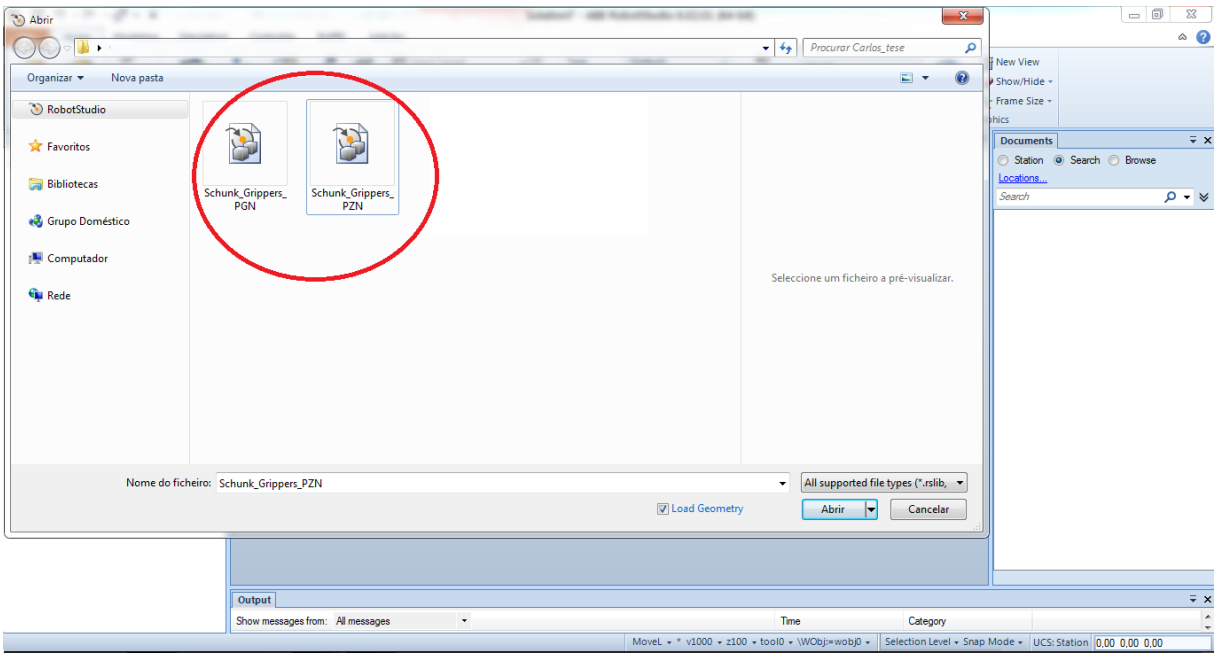
Smart Components

Para usar as garras de forma isolada numa estação desenvolvida, pode seguir os passos apresentados.

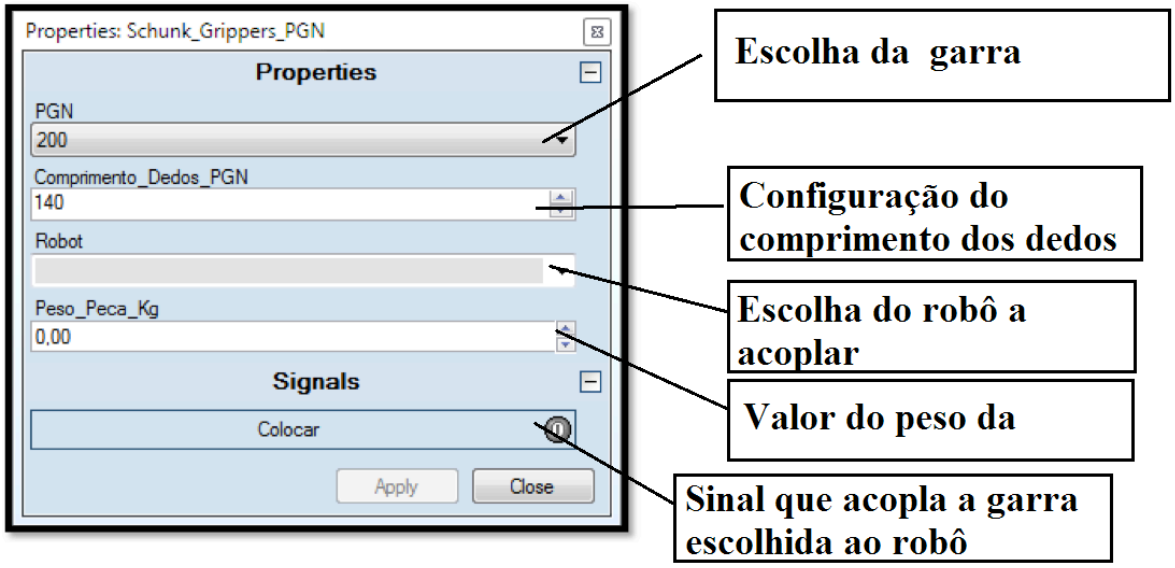
1- Importar o SC- Na aba *Import Library*, escolher *Browse for library* e indicar o diretótio onde se encontram os SC.



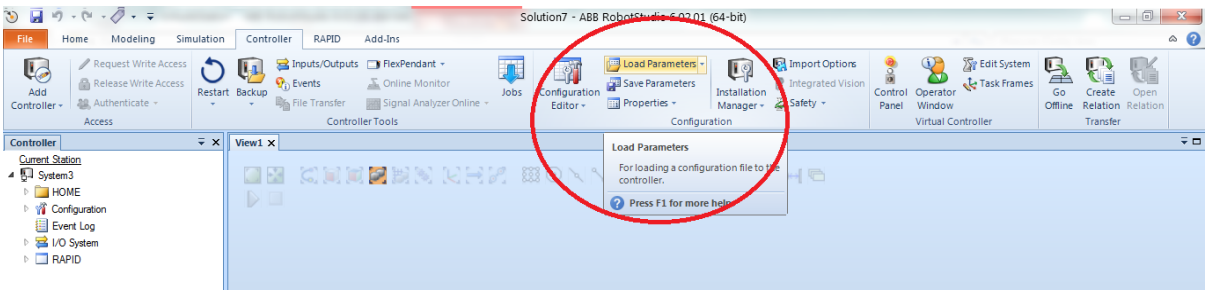
2- Escolha da série- Escoher um dos ficheiros .rslib representate de cada série de garras.



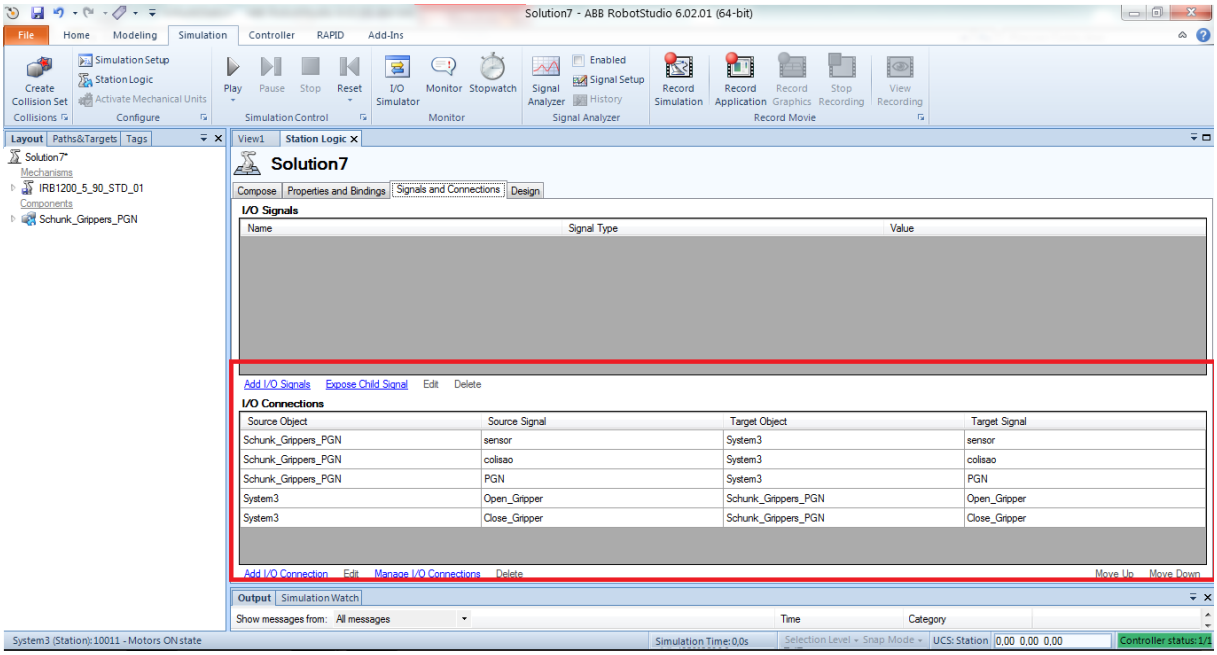
3- Janela do Smart Component - Com o botão do lado direito do rato abrir as propriedades do SC.



5- Parâmetros do controlador - Importar o ficheiro com as configurações na aba *Controller*, na opção *Load Parameters* através do ficheiro ".cfg".



6- Ligações ao controlador - Na aba de simulação criar as seguintes conexões entre o SC e o controlador.

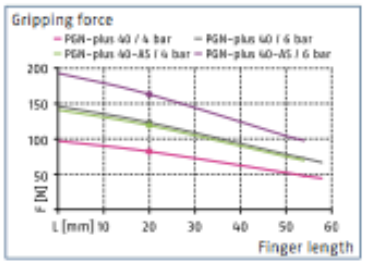


ANEXO B: *Datasheet* das séries de garras PGN-plus 40 e PZN-plus 40

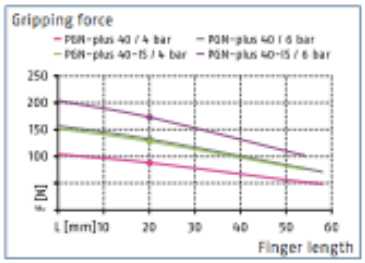
PGN-plus 40

SCHUNK Grippers pneumatic | 2-Finger Parallel Grippers | Universal Gripper

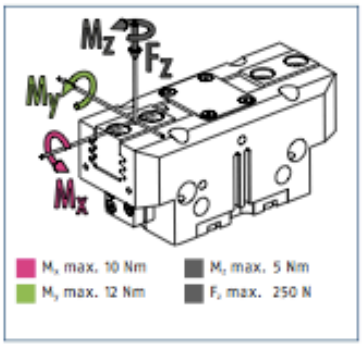
Gripping force, O.D. gripping



Gripping force, I.D. gripping



Finger load



① The indicated moments and forces are static values, apply per base jaw and may occur simultaneously. M_y may arise in addition to the moment generated by the gripping force itself. If the max. permitted finger weight is exceeded, it is impreative to throttle the air supply so that the jaw movement occurs without any hitting or bouncing. Service life may be reduced.

Technical data

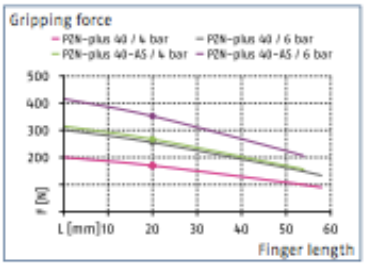
Description		PGN-plus 40	PGN-plus 40-AS	PGN-plus 40-IS
ID		0371080	0371082	0371084
Stroke per jaw	[mm]	2.5	2.5	2.5
Closing- / opening force	[N]	123/132	163/-	-/182
min. spring force	[N]		40	50
Weight	[kg]	0.08	0.1	0.1
Recommended workpiece weight	[kg]	0.62	0.62	0.62
Fluid consumption per double stroke	[cm³]	2.5	4.5	5.5
min. / max. operating pressure	[bar]	2.5/8	4/6.5	4/6.5
Nominal operating pressure	[bar]	6	6	6
Closing- / opening time	[s]	0.02/0.02	0.02/0.03	0.03/0.02
Closing- / opening time only with spring	[s]		0.05	0.05
max. permitted finger length	[mm]	58	54	54
max. permitted weight per finger	[kg]	0.1	0.1	0.1
IP class		40	40	40
min. / max. ambient temperature	[°C]	5/90	5/90	5/90
Repeat accuracy	[mm]	0.01	0.01	0.01
Cleanroom class ISO 14644-1		5	5	5
Options and their characteristics				
Dust-tight version		37371080	37371082	37371084
IP class		64	64	64
Weight	[kg]	0.1	0.12	0.12
Anti-corrosion version		38371080	38371082	38371084
High-temperature version		39371080	39371082	39371084
min. / max. ambient temperature	[°C]	5/130	5/130	5/130
Force intensified version		0372098	0372398	0372458
Closing- / opening force	[N]	225/235	265/-	-/285
Weight	[kg]	0.11	0.13	0.13
Maximum pressure	[bar]	6	6	6
max. permitted finger length	[mm]	50	50	50
Precision version		0371120	0371420	

① The full gripping force according to the data table is only realised after around 100 gripping cycles.

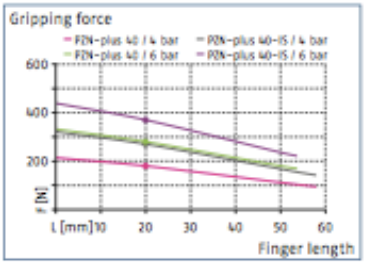
PZN-plus 40

SCHUNK Grippers pneumatic | 3-Finger Centric Grippers | Universal Gripper

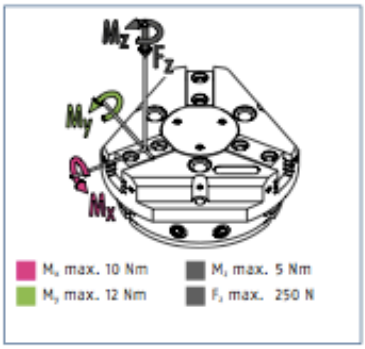
Gripping force, O.D. gripping



Gripping force, I.D. gripping



Finger load



① The indicated moments and forces are static values, apply per base jaw and may occur simultaneously. M_z may arise in addition to the moment generated by the gripping force itself. If the max. permitted finger weight is exceeded, it is impreative to throttle the air supply so that the jaw movement occurs without any hitting or bouncing. Service life may be reduced.

Technical data

Description		PZN-plus 40	PZN-plus 40-A5	PZN-plus 40-IS
ID		0303308	0303508	0303538
Stroke per jaw	[mm]	2.5	2.5	2.5
Closing- / opening force	[N]	255/270	355/-	-/370
min. spring force	[N]		100	100
Weight	[kg]	0.13	0.15	0.15
Recommended workpiece weight	[kg]	1.3	1.3	1.3
Fluid consumption per double stroke	[cm³]	5	9	9
min. / max. operating pressure	[bar]	2/8	4/6.5	4/6.5
Nominal operating pressure	[bar]	6	6	6
Closing- / opening time	[s]	0.03/0.03	0.02/0.04	0.04/0.02
Closing- / opening time only with spring	[s]		0.08	0.08
max. permitted finger length	[mm]	58	54	54
max. permitted weight per finger	[kg]	0.1	0.1	0.1
IP class		40	40	40
min. / max. ambient temperature	[°C]	5/90	5/90	5/90
Repeat accuracy	[mm]	0.01	0.01	0.01
Cleanroom class ISO 14644-1		5	5	5
Options and their characteristics				
Dust-tight version		37303308	37303508	37303538
IP class		64	64	64
Weight	[kg]	0.16	0.18	0.18
Anti-corrosion version		38303308	38303508	38303538
High-temperature version		39303308	39303508	39303538
min. / max. ambient temperature	[°C]	5/130	5/130	5/130
Force intensified version		0372199	0372219	0372239
Closing- / opening force	[N]	410/432	510/-	-/532
Weight	[kg]	0.19	0.21	0.21
Maximum pressure	[bar]	6	6	6
max. permitted finger length	[mm]	50	40	40
Precision version		0303338	0303488	

① The full gripping force according to the data table is only realised after around 100 gripping cycles.